

# Collaborating with designers

# What is design?

**“Design is deciding how a thing  
should be.”**

**— William Van Hecke**

# human-centered design

*noun*

A way of solving problems that centers the experiences and needs of humans throughout the design process.

**Not to be confused with...**

# software design

*noun*

**An engineering discipline of architecting software to meet product requirements in a flexible and scalable way.**

Both kinds of design are concerned with how a thing should be. We are focusing on **human-centered design**.

**For the purposes of today's  
conversation, a single definition:**



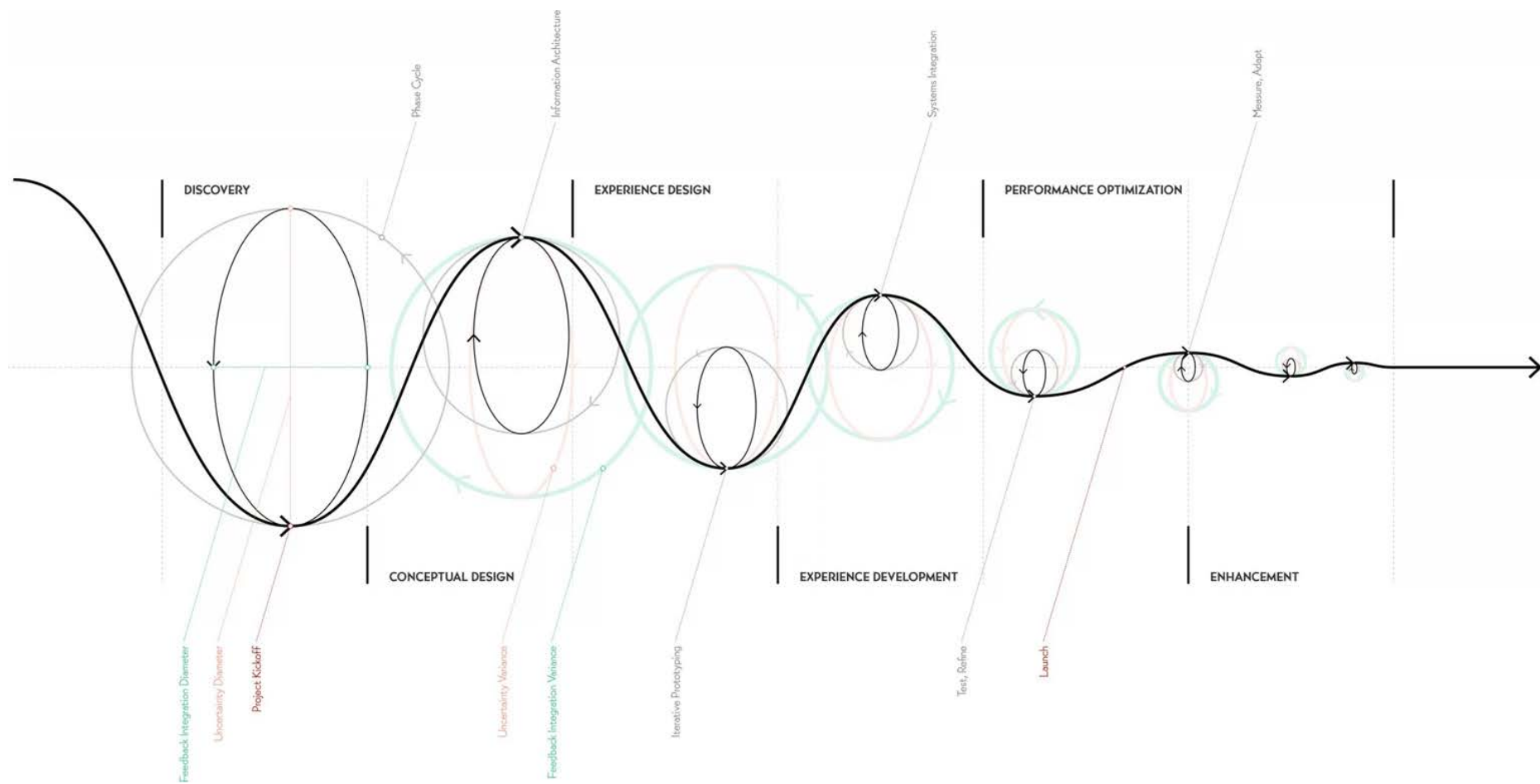
# designer

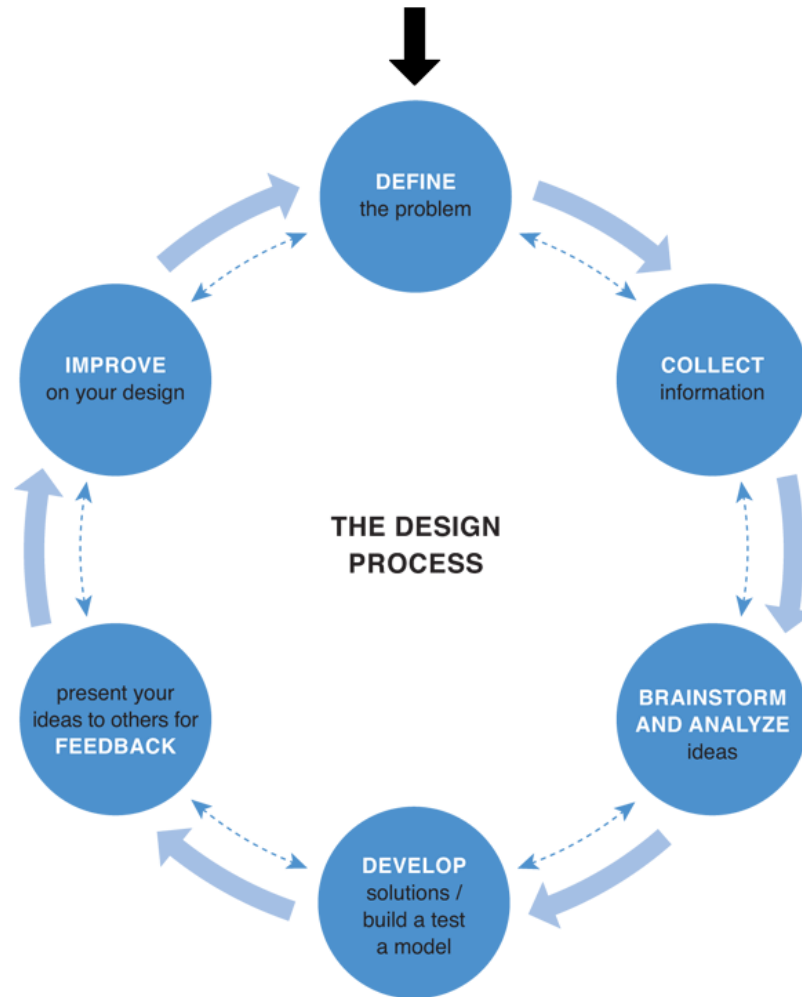
*noun*

Someone who uses knowledge of human experience to shape the experience of interacting with a particular product or service.

# The design process

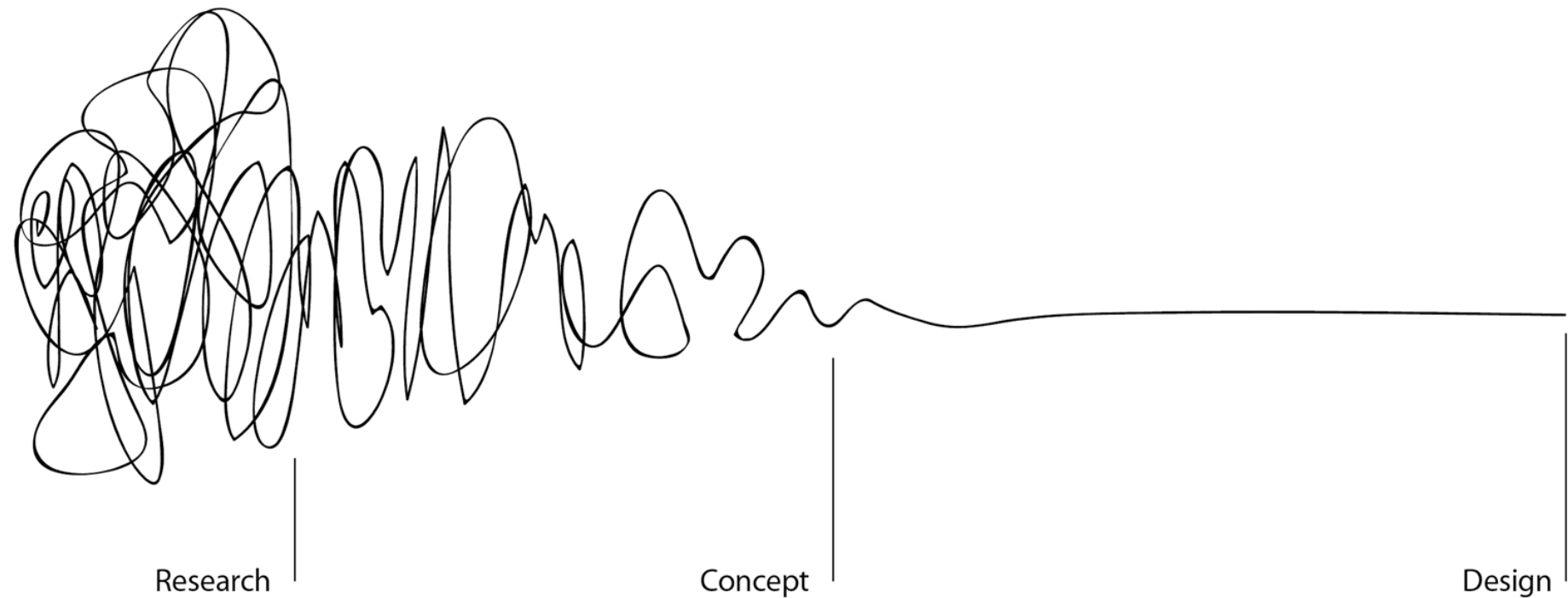
**What process do designers use?**





Uncertainty / patterns / insights

Clarity / Focus



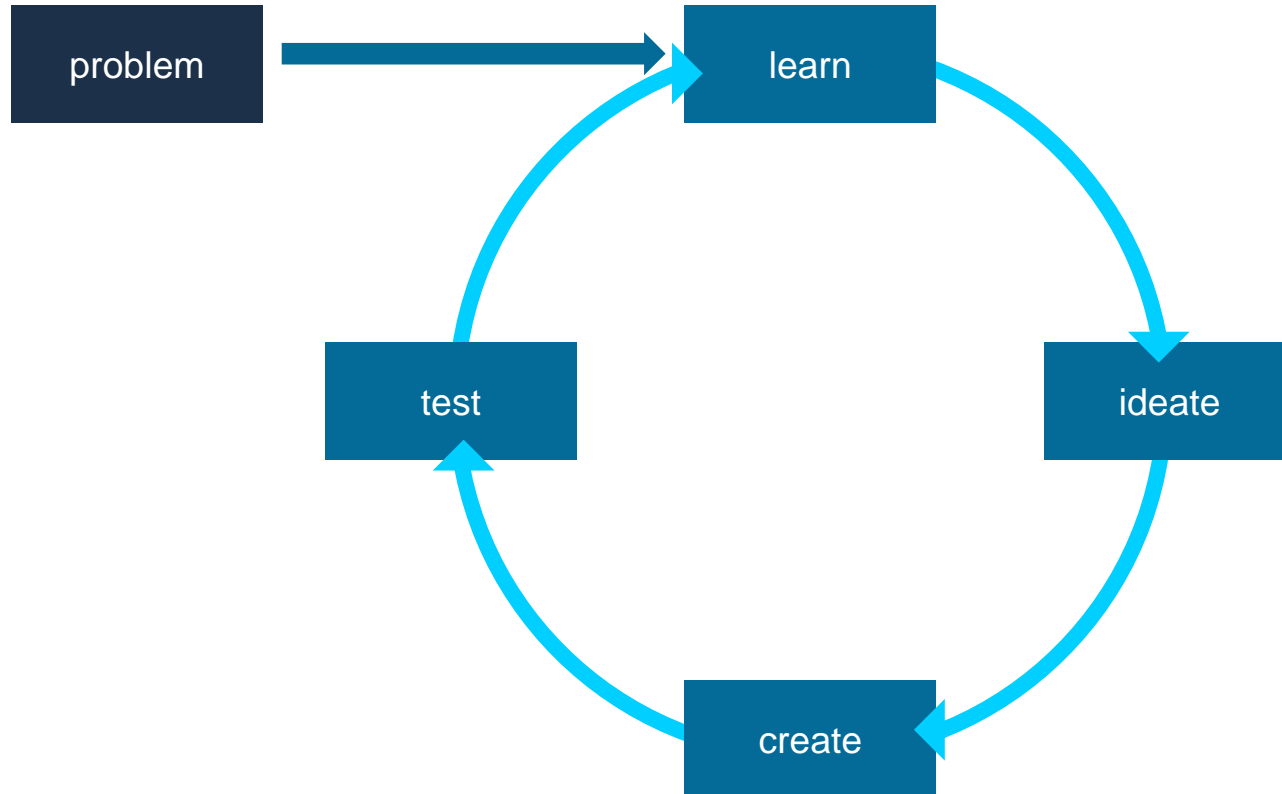
**There's no one true design process.**

**There's no one true design process...**  
**but there are consistent patterns.**



The design process

# A very oversimplified design process



# What do designers do?

**There are a lot of types of design,  
and designers have all sorts of  
specializations.**

What do designers do?

## A few common types of design

**User researchers** study people to understand their lives, contexts, and interactions with existing systems.

**Visual designers** specialize in visual communication — using the way things *look* to convey information and influence interpretation.

**UX designers** (and interaction designers) shape how people can interact with systems, and how those systems respond.

**Content designers** are experts in using language to convey ideas to people, most commonly in written form.

**Designers are often asked to wear many hats, and many are skilled at multiple kinds of design.**

**Someone whose résumé says “UX Designer” may also have great user research and visual design skills.**

Someone whose resume says “UX Designer” may also have great user research and visual design skills.

**They won't outdo a dedicated expert in those areas, but they can pinch hit comfortably.**

What do designers do?

## A few things designers do

- Discover what users **want** a system to do
- Discover what users **need** a system to do
- Help the team understand users' underlying goals and human motivations
- Create ideas for new products or services
- Sketch ideas for new products and services to help others understand them
- Design systems that are easy for users to understand and learn
- Create interactions that lead users to a desired action or thought process
- Prioritize users' needs
- Run usability tests to understand how well users can understand and use a system
- Use expert analysis to identify usability problems before testing
- Make it easy for users to notice, and fix, errors and bad data
- Create prototypes to test ideas before investing lots of development effort
- Shape how systems look and act to convey a specific personality, tone, or atmosphere
- Shape how language is used to convey a specific personality, tone, or atmosphere



What do designers do?

## A few things designers don't do

- Ensure the product vision is fulfilled
- Confirm your preconceived notions, even when they're inaccurate
- Make the design "pop more"
- Make the design "pretty"
- Draw the ideas in your head
- Read minds
- Research every single decision, big or small
- Add UI elements because they were asked to
- Ignore end user needs
- Follow numbers blindly
- Work alone
- "Throw things over the wall" to developers
- Fix the database
- Architect software
- Make the words sparkle
- Prioritize work
- Fetch coffee

**Design is not product, but good design is essential to the success of a product.**

# What design team do you want?

1

Building a system that will help city residents use the bus system more easily, with the goal of increasing ridership.

2

Deciding what to do with a statewide repository of public school statistics and performance data.

3

Publicizing an important new piece of policy about voter registration and identification to eligible voters in your county.

# Working with designers

Ask questions  
instead of...  
jumping to conclusions.

Learn what each designer  
specializes in (and what their  
secondary skills are)

instead of...

assuming all designers have the  
exact same skills.

Integrate research and design into  
your strategy

instead of...

ignoring design input *or* blindly  
following what designers say.

Define the outcomes you want  
instead of...

asking designers to implement a  
predefined solution.



Advocate for user research  
instead of...  
relying on assumptions about your  
end-users.

Explore multiple approaches  
instead of...  
doubling down on a single solution  
early.

Test designs with end-users  
instead of...  
arguing about which design is best.

Identify measurable success criteria  
instead of...  
choosing the design you “like” the  
most.

Create space and time for them to  
do good work  
instead of...  
rushing them.

Maintain forward momentum  
instead of...  
leaving designers to explore forever.

Be transparent about project  
priorities and tradeoffs  
instead of...  
forcing designers to guess.

**Keep track of the project schedule  
instead of...  
forcing designers to set their own  
deadlines.**



Provide direction on experience and  
implementation tradeoffs

instead of...

letting designers and engineers  
“work it out.”

# Practice

**Imagine you've been tasked with publicizing an important new piece of policy about voter registration and identification to eligible voters in your county.**

**You have a small team: one user researcher and one content designer. Both of them are moderately skilled at UX design.**

**What is your plan for working with them? What strategic questions do you need to answer together? Where are you hoping they will lead?**

# Collaborating with engineers

# What is engineering?

**Engineering is the practice of building systems that behave in a predictable, controlled manner.**



# software engineer

*noun*

**Someone who architects software to meet product requirements in a flexible and scalable way.**

**Not to be confused with...**

# electrical engineer

*noun*

Someone who uses knowledge of electricity, electronics, and electromagnetism to build controlled electrical systems.

# computer engineer

*noun*

Someone who uses knowledge of electrical engineering and software systems to develop new computing systems.

**\***

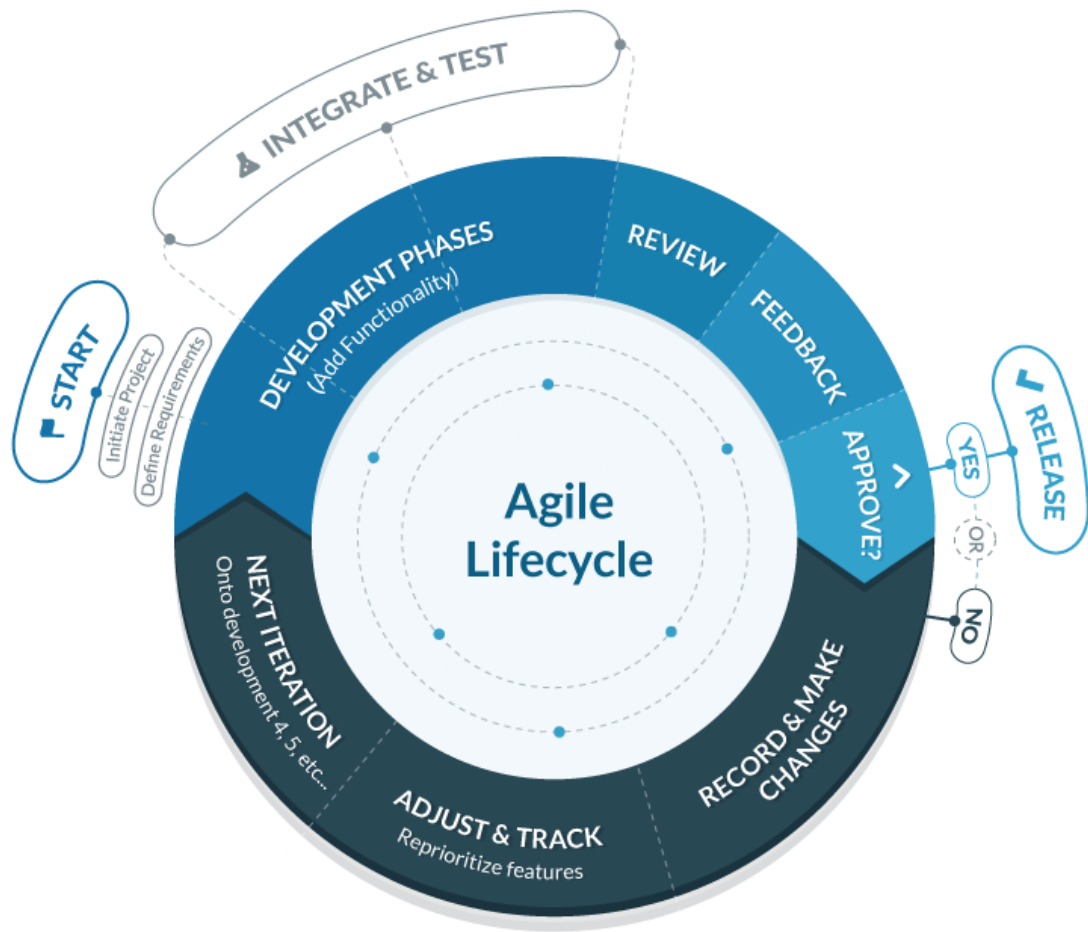
**Most engineers know how program computers, but very few of them know how to build flexible and scalable software systems.**

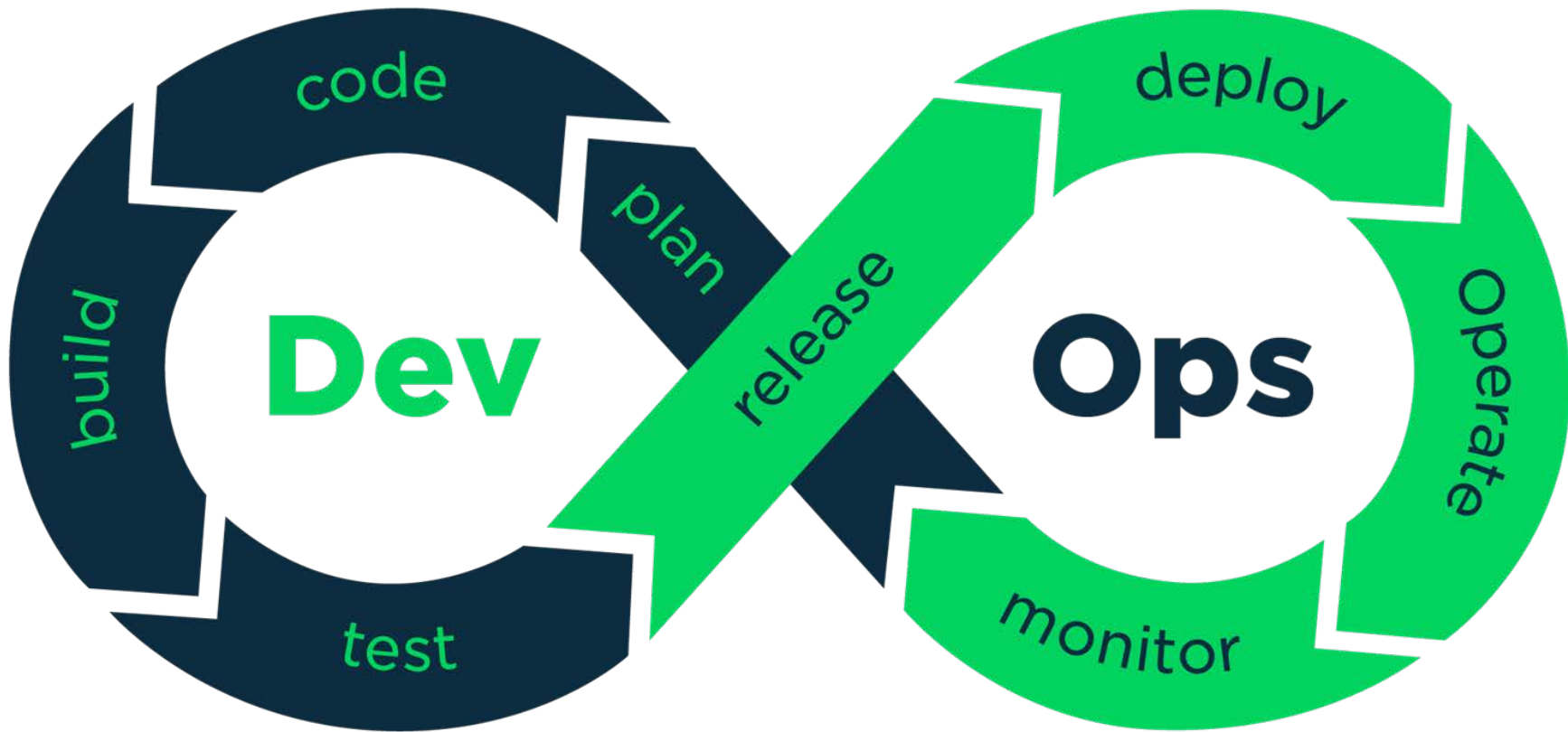
# The design process

**What process do software engineers use?**







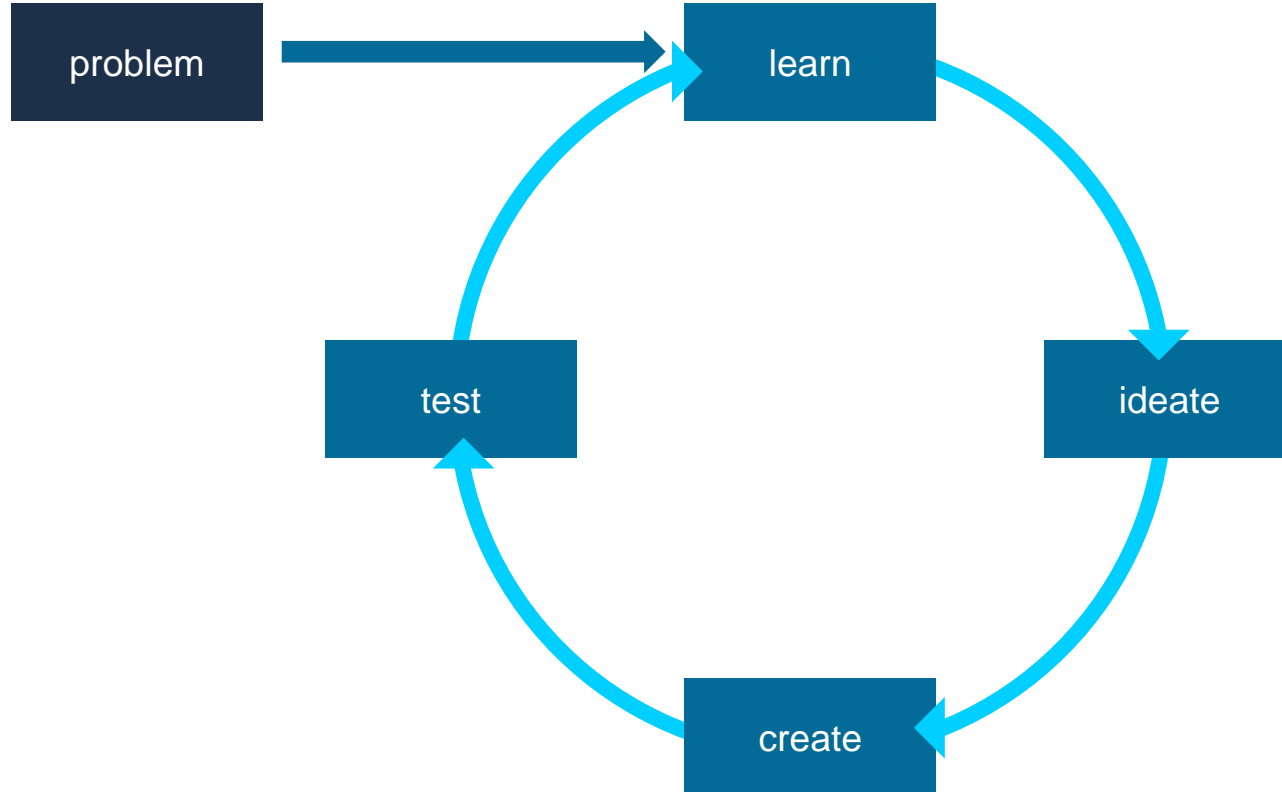


**There's no one true software development process.**

**There's no one true software  
development process...  
but there are consistent patterns.**

The design process

# A very oversimplified development process



# What do software engineers do?

What do software engineers do?

## A few common types of software engineers

**Front-end developers** build flexible, scalable UI systems that end-users directly interact with.

**Full-stack developers** build flexible, scalable UI systems and the back-end systems that manage the data users interact with

**Technical architects** coordinate the structural design of software systems to ensure that they are performant, scalable, and reliable.

**Back-end developers** build flexible, scalable software systems that manage the data that end-users create and interact with.

**Data engineers** wrestle with problems associated with database integration and messy, unstructured data sets so that users have access to data that works.

**DevOps engineers** create and maintain systems that allow software to be deployed, scaled, and monitored in an automated way.

**Many software engineers are skilled at multiple kinds of development.**



**For example, full-stack engineers are comfortable building complex infrastructure as well as building out the parts of the system that users directly interact with.**

What do software engineers do?

## A few things software engineers do

- Assess whether or not it's possible to build an idea
- Design a database schema
- Model business rules
- Optimize the performance of a system
- Choose an appropriate software framework
- Choose appropriate software tools
- Understand the security vulnerabilities of a system
- Review other engineers' code
- Help product managers prioritize work
- Create prototypes
- Implement continuous integration tests
- Write unit tests
- Implement telemetry and remote measurement tools
- Write systems documentation
- Make architectural decisions
- Use automated tools to test whether or not a system is accessible
- Use automated tools to test whether or not a system has security vulnerabilities
- Refactor code

What do software engineers do?

## A few things software engineers don't do

- Drive the product vision
- Design user research studies
- Design the look and feel of the system
- Prioritize work
- Whatever you say
- Define features
- Write copy
- Ignore technical constraints
- Follow numbers blindly
- Fetch coffee
- Work in a silo
- QA their own work

# What development team do you want?

1

Building a system that will help city residents use the bus system more easily, with the goal of increasing ridership.

2

Deciding what to do with a state-wide repository of public school statistics and performance data.

3

Publicizing an important new piece of policy about voter registration and identification to eligible voters in your county.

# Working with software engineers

Ask questions  
instead of...  
jumping to conclusions.

Learn what each engineer  
specializes in (and what their  
secondary skills are)

instead of...

assuming all engineers have the  
exact same skills.

Integrate technological  
considerations

instead of...

ignoring technical input *or* blindly  
following what engineers say.



Define the outcomes you want  
instead of...

asking engineers to implement a  
predefined solution.

**Be clear about acceptance criteria**  
**instead of...**

**letting the engineers figure out when  
a user story is 'done'.**

Prototype concepts

instead of...

doubling down on a single solution  
early.

Encourage experimentation  
instead of...  
insisting on perfection.

Identify measurable success criteria  
instead of...  
choosing the implementation you  
“like” the most.

Invest in automation and  
architecture  
instead of...  
cutting corners to save time early on.

Create space and time for them to  
do good work  
instead of...  
rushing them.

Maintain forward momentum  
instead of...  
leaving engineers to optimize  
forever.



Be transparent about the balance  
between speed and quality  
instead of...  
forcing engineers to guess.

Clearly communicate milestones  
instead of...  
forcing engineers to set their own  
deadlines.

Provide direction on experience and  
implementation tradeoffs

instead of...

letting designers and engineers  
“work it out”.

**Include engineers in discussions  
about vision and strategy**

**instead of...**

**assuming they won't care or have  
good input.**

Expose engineers to real users and  
their problems  
instead of...  
'protecting' them from distractions.

Include refactoring when you  
consider priorities  
instead of...  
letting technical debt build up.

Show them the value they're creating  
for users  
instead of...  
assuming they don't care.

Encourage a culture of collaboration,  
communication and knowledge  
sharing  
instead of...  
enabling silos.



# Practice

**Imagine you've been tasked with publicizing an important new piece of policy about voter registration and identification to eligible voters in your county.**

**You have a small team: two full-stack developers.**

**What is your plan for working with them? What strategic questions do you need to answer together? Where are you hoping they will lead?**

# Leading across disciplines

# Balance

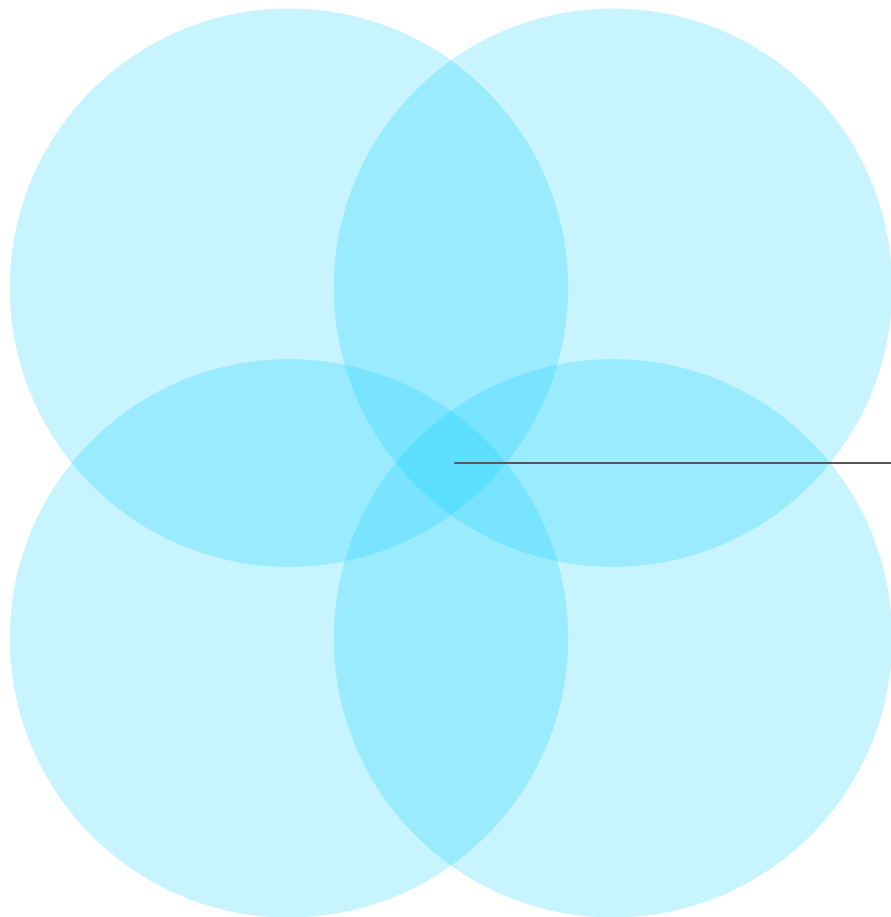
user need

stakeholders

**great  
product**

technology

business



**Everyone loses when the product strategy is out of balance.**



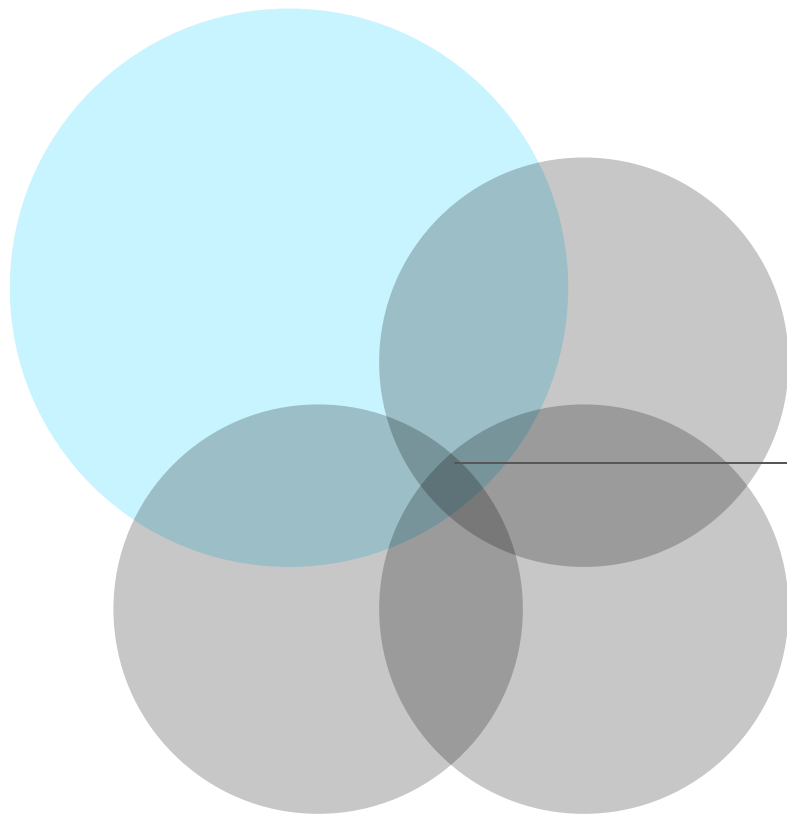
user need

stakeholders

**shaky  
foundation**

technology

business



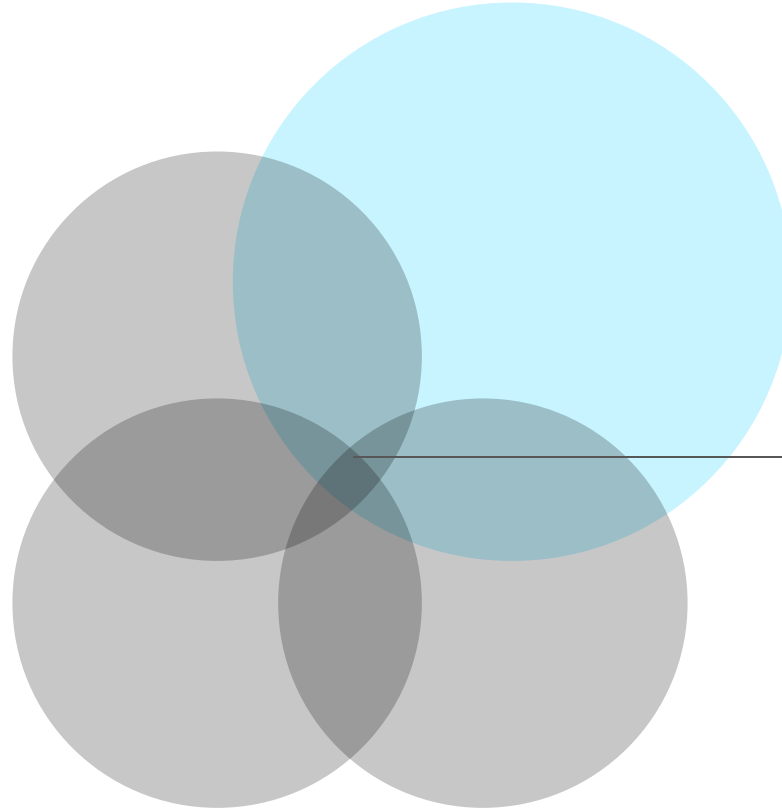
user need

stakeholders

**waste of  
resources**

technology

business



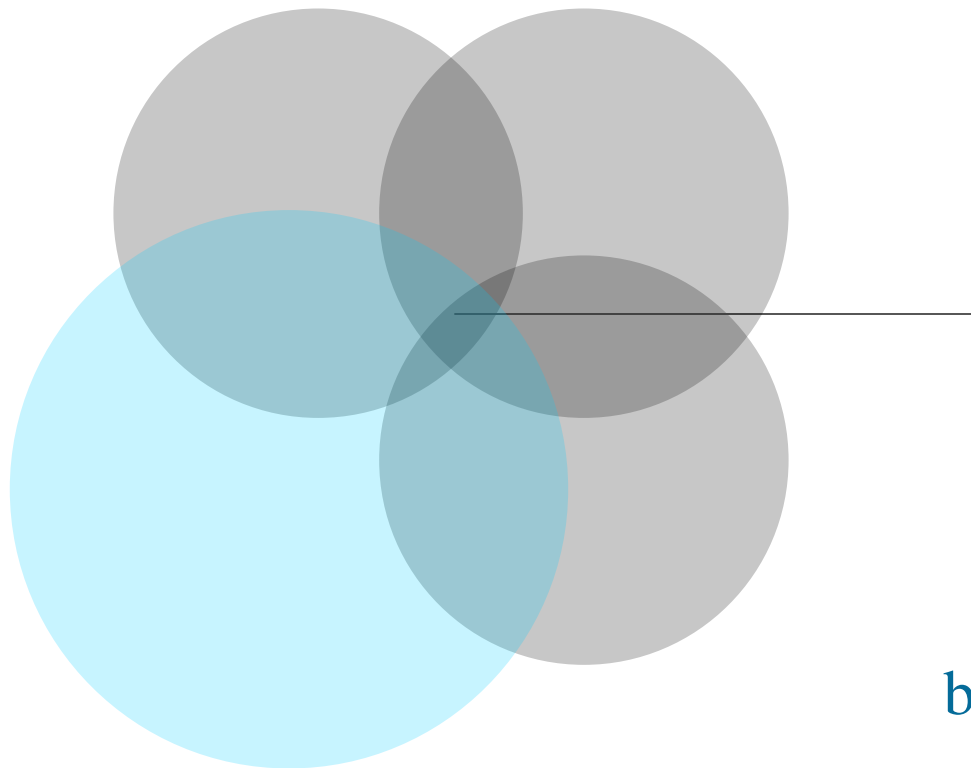
user need

stakeholders

doubtful  
value

technology

business



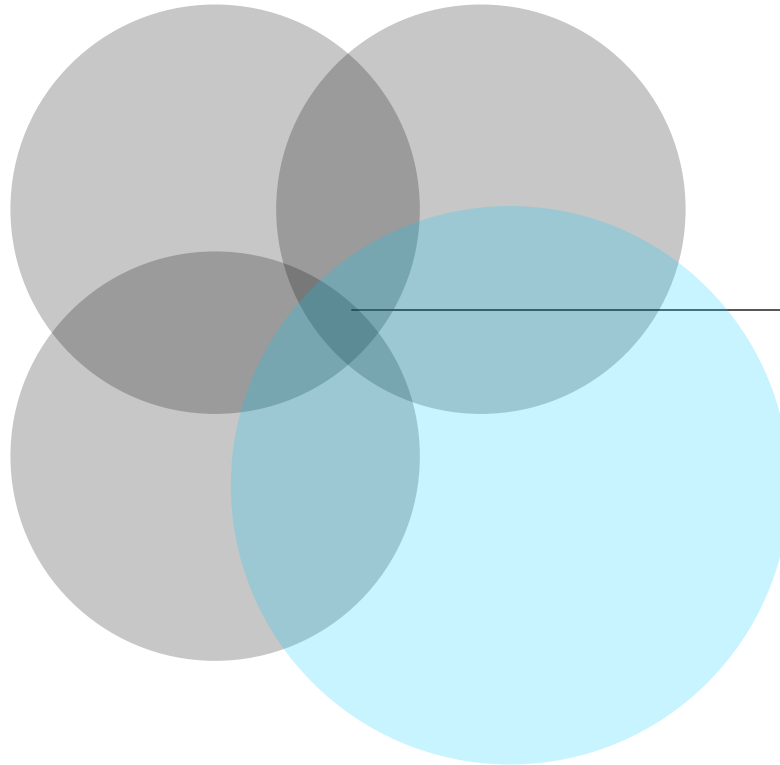
user need

stakeholders

**impossible  
to sell**

technology

business



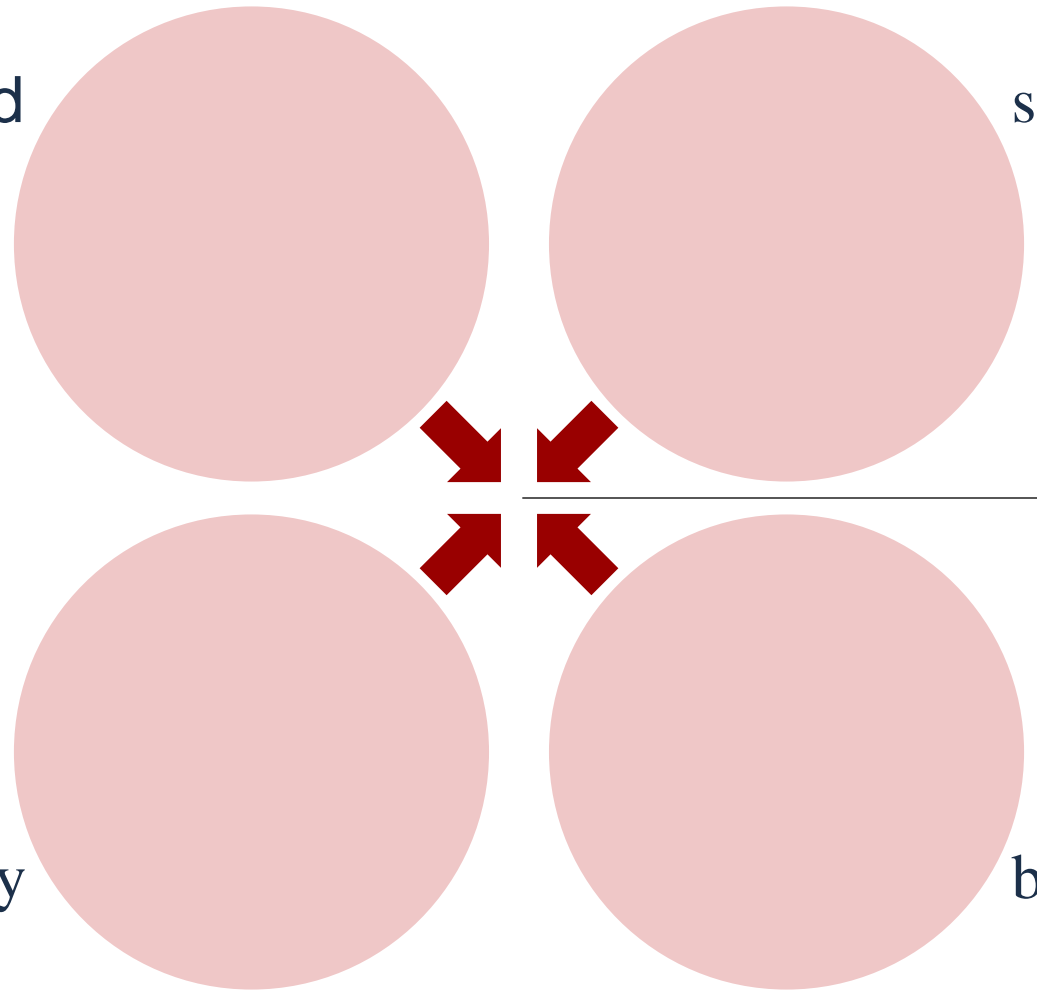
user need

stakeholders

total  
chaos

technology

business



**You have to find the right way to reconcile all of the different inputs and make sensible tradeoffs.**

You have to find the right way to reconcile all of the different inputs and make sensible tradeoffs. **Your vision statement should help.**

# Driving from the vision



**Your vision statement should acknowledge all of the forces affecting your project. This creates a central point of alignment that you can use to resolve tough conversations as you work.**

**Creating alignment is work.**

Even when everyone agrees on the vision, you still have to drive alignment on **how to get there.**

# Driving alignment



Articulate the goal explicitly



Show your work loudly and often — demos over memos



Create objective success metrics



Groom your narrative regularly

# Practice

**Imagine you're building an analytics platform to track how satisfied constituents are with local Department of Transportation services.**

**You've worked hard to get everyone aligned around this effort, but your key executive sponsor has just retired. The new executive says it's a waste of money, because "we already know that the public wants new roads and hates construction."**

**What do you do?**



**Your design team is really excited about creating a mobile app where members of the public can report transit problems (like potholes), track the status of their past reports, and see how involved they are compared to other residents.**

**In preliminary usability tests, multiple users have told you that the design is easy to use, but that they don't want to be compared to their neighbors.**

**And the department head is uncomfortable with allowing users to track reported issues in detail, because delays related to funding, weather, and other external forces make the department look unresponsive.**

**What do you do?**

**The city's Chief Data Officer has told you that the data team can do much more valuable analysis if your app tracks all of the reports about a particular problem over time (longitudinal history) in addition to individual reports.**

**Your dev team has told you that tracking and correlating all of the reports about a single transit problem will take them at least 8 months to deliver. The Chief Data Officer argues that it shouldn't take them nearly that long.**

**What do you do?**

**You're trying to create a simple interactive app to test some of the new concepts the design team has come up with.**



**Your dev lead has estimated several months of development time to do this, and when you ask why, they explain that there is a lot of infrastructure that the team will need to build to support this experience properly.**

**Your dev lead argues that the team will have to do the work eventually to build out the app, and that it's better to create scalable infrastructure now so that the team won't have to refactor it later.**

**What do you do?**

**You are working with a client who has a dataset around health outcomes by region and they want to create a visualization of the data for the public**

**Your primary stakeholder has very clear ideas about what data to show and how the data is visualized. They “already know what users want” so they don’t want to spend time doing research**

**Your dev lead has already planned a data structure based on the primary stakeholder's vision**

**What do you do?**

# Legacy Modernization Basics



# What is a legacy system?

**In the software world, legacy systems tend to be big, complex systems that are hard to change and expensive to maintain**

**Many times their design is not well understood, and they frequently fail to meet the current or emerging needs of their users**

**These systems tend to grow out of legacy structures and processes that were not designed to cope with the pace at which the world changes**

**\***

**Legacy systems can be 30 years old, or  
something you just bought off the shelf**

# How do you know your system is legacy?

**You are working with technology or hardware that is hard to support and maintain**

**As time goes by, the costs of your system increase as benefits decrease for your workers and beneficiaries**



**You don't know why system workflows  
are the way they are, and they interfere  
with users trying to accomplish their goals**

**The software is not adaptable to changes in the business - if business requirements change, it's hard or impossible for the software to change**

**The software is not adaptable to changes in technology (mobile, artificial intelligence, data science), or changes in the usage patterns**

**You are 'locked-in' with a specific vendor  
because you cannot fix or iterate on the  
software without that vendor's help**

**It's hard or impossible to get accurate and useful data from your system because of a lack of data accuracy or integrity across the system**

# What is legacy modernization?

**Legacy modernization is a way of thinking and working that is contrary to the lengthy traditional process of design, develop, install, operate and maintain.**

**It involves continually evolving or replacing outdated systems and processes in order to improve outcomes for all constituents**



**While also reducing costs, improving  
system flexibility, and improving  
security posture**

**In this world ...**

**Software is never in maintenance mode, and is always being improved, so that you are always in a position to respond to change**

**In this world ...**

**Your IT organization is optimized for quick, continuous delivery of value to users, rather than month long deployment cycles**

**In this world ...**

**Data is designed to be consistent and meaningful across the entire system, and the right people have access to the right data**

**In this world ...**

**Procurements are smaller, and more specific, and there is a variety of non-traditional vendors competing for the work**

**In this world ...**

**Security is addressed incrementally,  
rather than at the very end of a lengthy  
(multi-year) development effort**

# Why should I care?

**Because technologies and user needs are changing fast, systems must change at the same rate or become legacy**



**Legacy systems cause a number of problems for the organization**

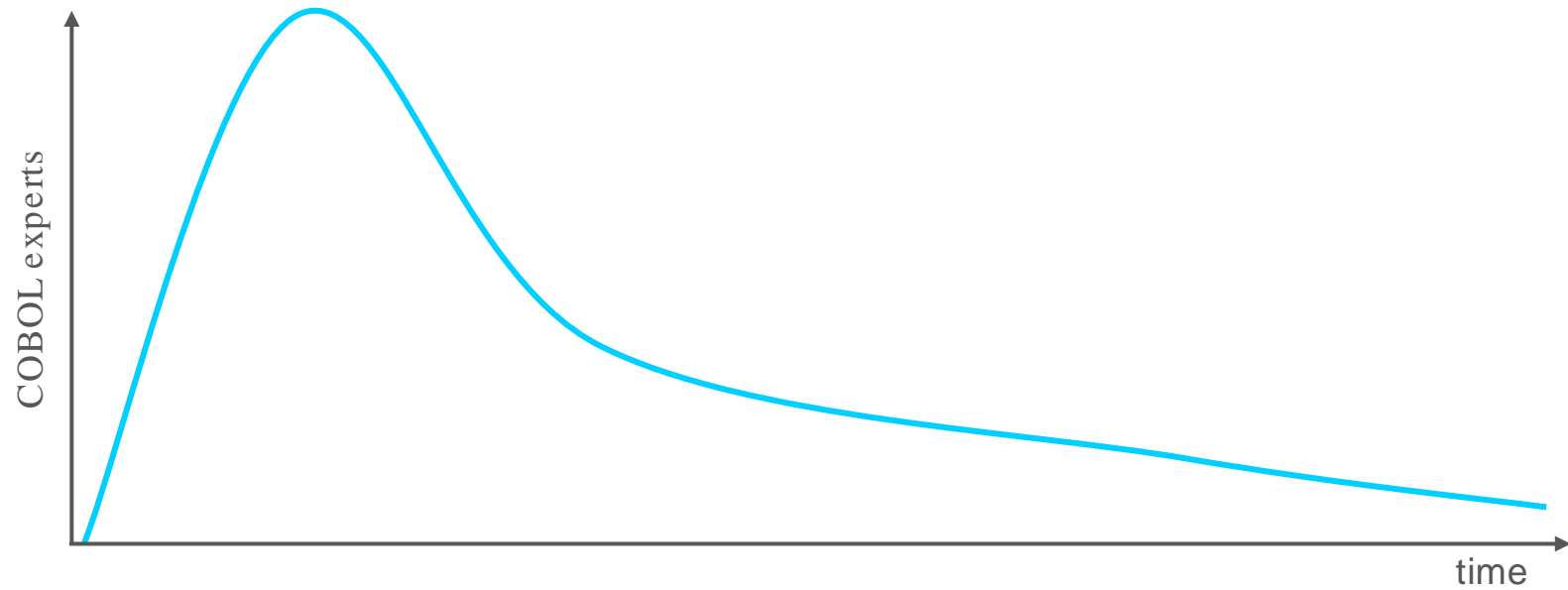
**Workers will be less productive as more and more manual workarounds get implemented to compensate for the system's failure to adapt**

**End users of your system will become frustrated and you will have to spend more time handling complaints than delivering on the mission**

**Your ability to iterate on the system will be trivial or non-existent, and this will prevent you from finding capable vendors to do the work**

**You will be unable to easily use the system data to comply with standard audits or custom requests, much less leverage the data for predictive models**

**You will have trouble hiring staff to operate and maintain the system as it degrades, and as expertise in old technology ages out**



**In general, legacy systems increase costs in dollars, time, resources and morale, as your ability to execute on your mission continues to diminish**



**Modernizing will give you a flexible and sustainable system and organization that is responsive to change**

**So you may better match your  
business needs and user needs at any  
given time**

**And so you can continuously and  
effectively deliver on your mission**

# How do I modernize my legacy system?

**The first step is to bring ownership of the system in-house and out of the hands of external vendors**

**Ownership should be at all levels of the organization from business to IT to security and ops**

**As a whole, the organization defines  
and measures the desired outcomes  
for all constituents**

**And actively holds vendors  
accountable for delivering those  
outcomes on a continual basis**



**To start, identify all of your constituents  
(users, stakeholders, partners)**

**Align on a vision of what success looks like for all constituents**

**Conduct research and workshops to identify your goals & priorities, and schedule them in a roadmap**

**Build or buy towards that roadmap  
incrementally and bring users along  
each step of the way**

**Be sure to stay connected with users  
(through research and feedback)  
throughout the entire migration**

**Communicate constantly with stakeholders, vendors, and end users throughout the process**

**Take steps to ensure that you can  
modernize continually after the initial  
migration**

**So that you don't end up with another  
legacy system in 5 years**



# What else needs to happen?

**To work this way, you'll need an organization, infrastructure and culture that embraces product thinking and adaptability**

**So that you can constantly identify and respond to changes in user needs, new technologies or new business requirements**

**Create a strong product practice at your organization to ensure that the product is cared for after the migration is complete**

**Maintain an up-to-date roadmap at all times informed by continual research into users, existing products and services, new technologies, and organizational or policy changes**

**Ensure that security concerns are dealt with in a way that does not block continuous delivery of value to your users**

# Managing product risks

**Every product effort has risks.  
Identifying, tracking and managing  
them from the beginning is critical to  
product success.**



# What is product risk?

**A product risk is anything that could get in the way of the team achieving the vision**

**Risks can be internal or external.**

Risks can be internal or external.  
**They can be around business,  
product, project or process.**

Risks can be internal or external. They can be around business, product, project or process. **They have varying impacts, and likelihoods.**

**Like anything else, risks change all the time, because the world changes**

Like anything else, risks change all the time, because the world changes. **So they need to be constantly assessed.**

**Some risks get riskier over time**



**Some risks subside over time**

**New risks can develop at any time**

**Actively managing risk is an important part of any product process**

# Managing risk

**Managing risk is a **practice** that helps minimize the impact of threats to the product**

Managing product risk

# The practice of risk management

1

Identify

2

Define

3

Prioritize

4

Mitigate and  
measure

5

Communicate

## Identify

**Don't wait for risks to find you. Keep an eye out for risks and identify and record them quickly**

## Define

**What is the nature of the risk? What is its potential impact? What is its likelihood?**

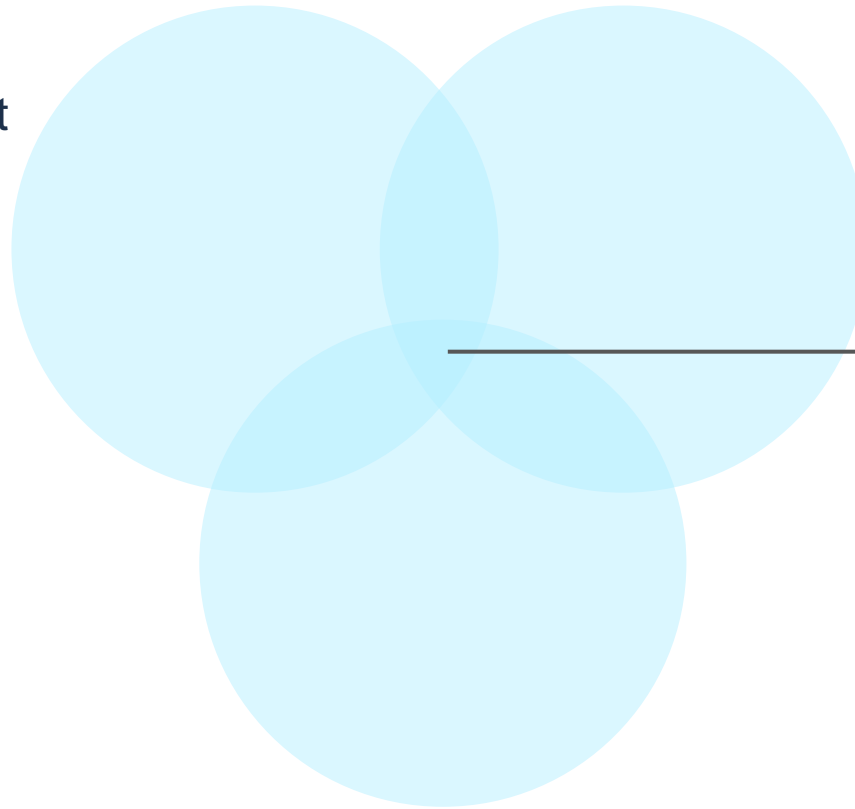


## Prioritize

**Which risks do you work on now?  
Which can wait?**

Risk requirement

Risk tolerance



Prioritize the risks that are required, that the team feels comfortable taking, and that you can afford

Financial capacity

## Mitigate and measure

**Come up with a plan to mitigate high priority risks and work it into your product backlog. Come up with metrics to measure your progress**

## Communicate

**Make sure everyone knows what you're doing to mitigate the high priority risks and what's working**

**Risks and their status need to be visible to everyone at all times.**

**Because they affect stakeholders,  
users and any constituent who  
hopes to get value from the product**

# Risk management as a feature

**It's tempting to want to avoid talking about risk because it scares people.**



**But in fact, risk management is already a focus of what product teams build and do. And, that's a good thing!**

**There is always a risk that security may be breached on a given system, which is why we implement authentication and authorization.**

**There is always a risk that working on large complex features will result in problematic implementations, which is why we break complex features into smaller ones.**

**There is always a risk that we will miss a critical stakeholder which is why we continually communicate our product story**

**Protecting your constituents from risk is valuable. So, this work should be called out and prioritized along with any other value you are trying to deliver**

**So, make a point to bring up risks and talk openly about them with your team and any stakeholders.**

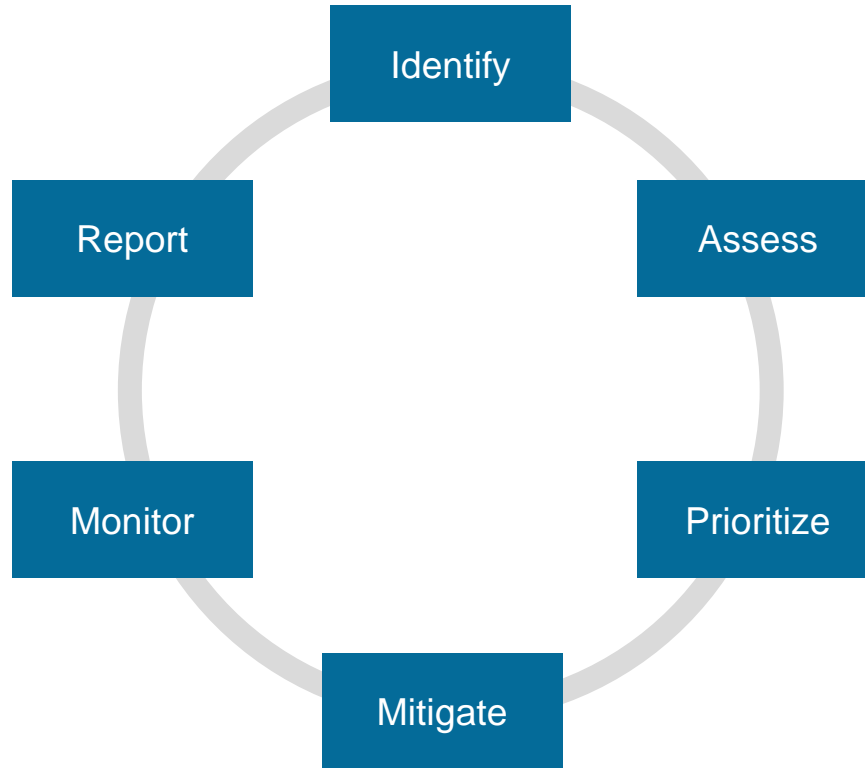
**Doing this protects your team and  
opens you up to innovation.**

# What are your product risks?



**End of presentation**

## Risk Management Cycle



**Doing this work will guide the team  
to an achievable and desirable end  
state**

**It will also reduce the impact of failures as everyone will already have aligned on the nature of the risk and why it was taken**

# Measurement and metrics

**Measurement is about creating a  
feedback loop.**

Measurement is about creating a feedback loop. **It gives you a way to understand how your product exists in the world.**

**You're constantly getting feedback about your product, whether or not you measure it consciously.**



You're constantly getting feedback about your product, whether or not you measure it consciously. **And the information you're getting is biased.**

**You can't prevent bias, but you can mitigate it by being careful about how you measure things — and how you interpret the data you get.**

# Using measurement

## Measurement helps you:

- \* find problems
- \* find opportunities
- \* prioritize resource use
- \* back up your product story

**Good measurement leads to action.**

\*

**That action is often “investigate \_\_\_\_”**

# Different types of measurement

1

**Performance metrics** tell you whether or not your product is working as intended.

2

**Business metrics** tell you how well your product supports your organization's goals.

3

**Exploratory metrics** tell you how people are using your product and help you understand where to go next.

**Imagine you're building a website  
where eligible individuals can  
register for SNAP benefits.**



## Performance metrics

Usability test  
performance

Abandonment  
during signup

Server uptime

Page load time

Performance metrics

Usability test  
performance

Abandonment  
during signup

57% of users bail out during  
signup. We should do some  
testing to find out why!

Server uptime

Page load time

Performance metrics

Usability test  
performance

Abandonment  
during signup

57% of users bail out during  
sign up. We should do some  
testing to find out why!

It's because they got  
frustrated with the process.  
We should redesign the flow.

Server uptime

Page load time

Performance metrics

Usability test  
performance

Abandonment  
during signup

57% of users bail out during  
sign up. We should do some  
testing to find out why!

It's because they had to find  
a way to prove their identity.  
We should investigate that.

Server uptime

Page load time

Performance metrics

Usability test  
performance

Abandonment  
during signup

57% of users bail out during  
sign up. We should do some  
testing to find out why!

It's because they figured out  
they were ineligible. *Signup*  
isn't the real problem.

Server uptime

Page load time

Performance metrics

Usability test  
performance

Abandonment  
during signup

Server uptime

Page load time

**Metrics don't tell you *why*.  
They just tell you something  
is happening.**

## Performance

Usability test  
performance

Abandonment  
during signup

Server uptime

Page load time

Performance

Usability test  
performance

Abandonment  
during signup

Server uptime

Page load time

Business

Time to complete  
enrollment

Enrollment system  
maintenance cost

Number of  
incomplete or  
inaccurate  
applications



## Performance

Usability test  
performance

Abandonment  
during signup

Server uptime

Page load time

## Business

Time to complete  
enrollment

Enrollment system  
maintenance cost

Number of  
incomplete or  
inaccurate  
applications

## Exploratory

SNAP program  
enrollment rates

SNAP utilization  
rates

**Sometimes a single metric  
fits into multiple categories.  
That's fine!**

Exploratory

SNAP program  
enrollment rates

SNAP utilization  
rates

Abandonment  
during signup

# Communicating with metrics

As long as your project is funded by  
**other people**, you will have to  
convince them that it's worthwhile.

# People like:

- \* your manager
- \* the legislature
- \* a venture capital firm
- \* customers

They want to know two things:

1. **Why should I care** about what you're trying to do?

# They want to know two things:

1. Why should I care about what you're trying to do?

2. **Why should I trust you to get it done?**

**Metrics help you answer both questions. Most people will accept them as a form of **proof**.**



\*

**It falls on you to use metrics responsibly. Acknowledge biases. Question your conclusions. Don't trust numbers blindly.**

**We're building a better SNAP  
enrollment system.**

**We're building a better SNAP  
enrollment system because 60% of  
eligible residents aren't signed up.**

**We're really happy with this design.**

**We're really happy with this design.**  
**Usability test participants finished enrollment about twice as quickly as with the existing system.**

**We're really happy with this design.**  
**Usability test participants completed**  
**signup without errors 90% of the**  
**time.**

We're really happy with this design.  
Usability test participants told us  
that they “love this new way of  
signing up — it's so easy”.

# Choosing the right metrics



**Your metrics should be based on  
your goals.**

Your metrics should be based on your goals. **They're how you know if you're meeting the goal or not.**

**Let's go back to our example:**  
**You're building a website where**  
**eligible individuals can register for**  
**SNAP benefits.**

**You probably have some goals, like**

You probably have some goals, like  
getting people enrolled more quickly.

You probably have some goals, like  
making life easier for analysts.

You probably have some goals, like  
reducing application errors.

**Each of those goals should be mapped to one or more metrics.**



## Goal

Get people enrolled more quickly.

Make life easier for analysts.

Reduce application errors.

## Goal

## Metric(s)

Get people enrolled more quickly.



Time to complete enrollment

Make life easier for analysts.



Number of incomplete or inaccurate applications

Reduce application errors.



Number of incomplete or inaccurate applications

Usability test performance

**Just as you've prioritized your goals,  
you should also prioritize your  
metrics.**

**The metrics that track your most important goals? They probably tell you the most about how your product is doing overall.**

Those are your **Key Performance Indicators** (often shortened to KPIs).

# Practice

**Imagine you're in charge of replacing the app that calculates retirement benefits for agency staff.**

**What are your goals for this project?**  
**What does success look like?**



**\***

**Create at least one business goal,  
one user experience goal, and one  
technical goal.**

**How can you measure whether or not you're meeting each of those goals?**

**What do those measures mean?**  
**If they spike or dip, what action will**  
**you take?**

# Organizing around value

# Agenda

1. What is the opportunity?
2. What is a value stream?
3. Organizing around value streams

# What is the opportunity?

**The TTS strategy effort is a great opportunity to make explicit the variety of ways we deliver on our mission**

**It will bring the PIF program, 18F,  
Acquisitions and OPP all under a single  
vision**



**And help us settle on an inclusive strategy that enables TTS to deliver the most value to our government and the public**

**That strategy should clearly articulate the mechanisms by which we produce value consistent with our mission**

**So that we can organize around the  
highest priority products and services  
that help us deliver that value**

The concept of **value streams** can help  
TTS think this through

# What is a value stream?

**Together, we already provide a great deal of value to the government and people of this country**

- Cloud services
- Search
- Consulting
- Custom projects
- Innovation lab
- Security services
- *There are probably more ...*

**Each of these can be thought of as a stream of value flowing to our users**



Starting with a **trigger** or fundamental need

**And ending with value being delivered  
to end users whether they be  
government or the public**

**If we identify all of the value streams**

**And we identify the services and products that support those value streams**

**We can organize around those that  
create the most impact**

**By allocating budget and staff towards  
achieving that impact**

**So that we can better deliver on our mission**

**And by articulating metrics, so that we can measure how we're doing and at regular intervals, and reassess our strategy**



**Going through this exercise together  
will help everyone understand clearly  
who we are and whether we are  
delivering on our mission**

**And we will all be able see much more  
clearly what the priorities are and why**

# Organizing around value streams

**Once we identify and prioritize value streams, it will be easier to see how to organize TTS around them**

# Staffing

# Hiring

# Operations

**End of presentation**





# Roadmapping

# The point of roadmaps

**Product roadmaps are a powerful strategic tool.**

Product roadmaps are a powerful strategic tool. **They help you keep track of the big picture and correlate every work item to your end goal.**

Product roadmaps are a powerful strategic tool. They help you keep track of the big picture and correlate every work item to your end goal.

**They also help you stay aligned with your stakeholders.**

You shouldn't use roadmaps to forecast exactly *what* will get delivered *when*.

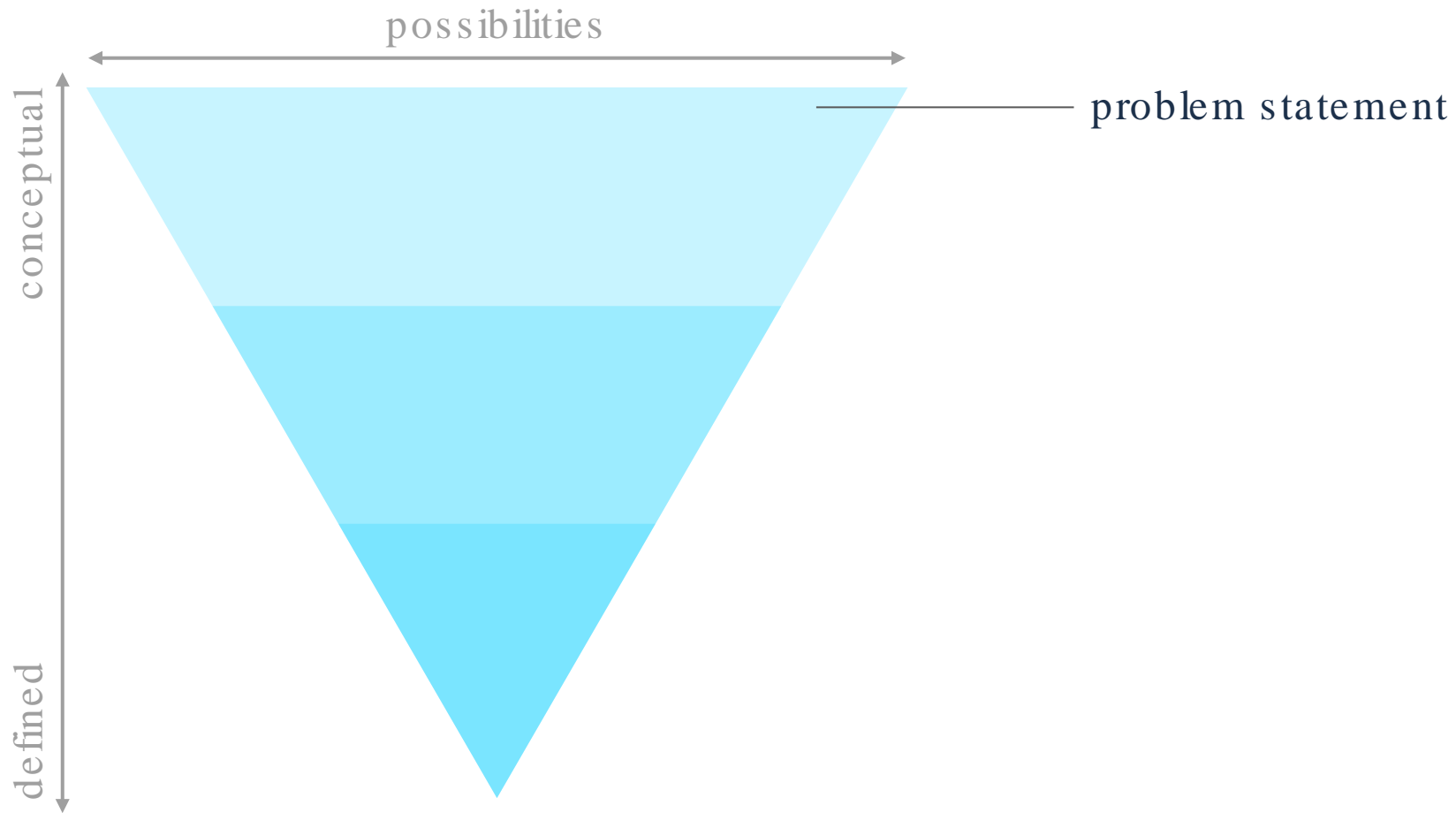
You shouldn't use roadmaps to forecast exactly *what* will get delivered *when*. **They're for planning the order in which you'll tackle the various pieces of the overarching problem.**

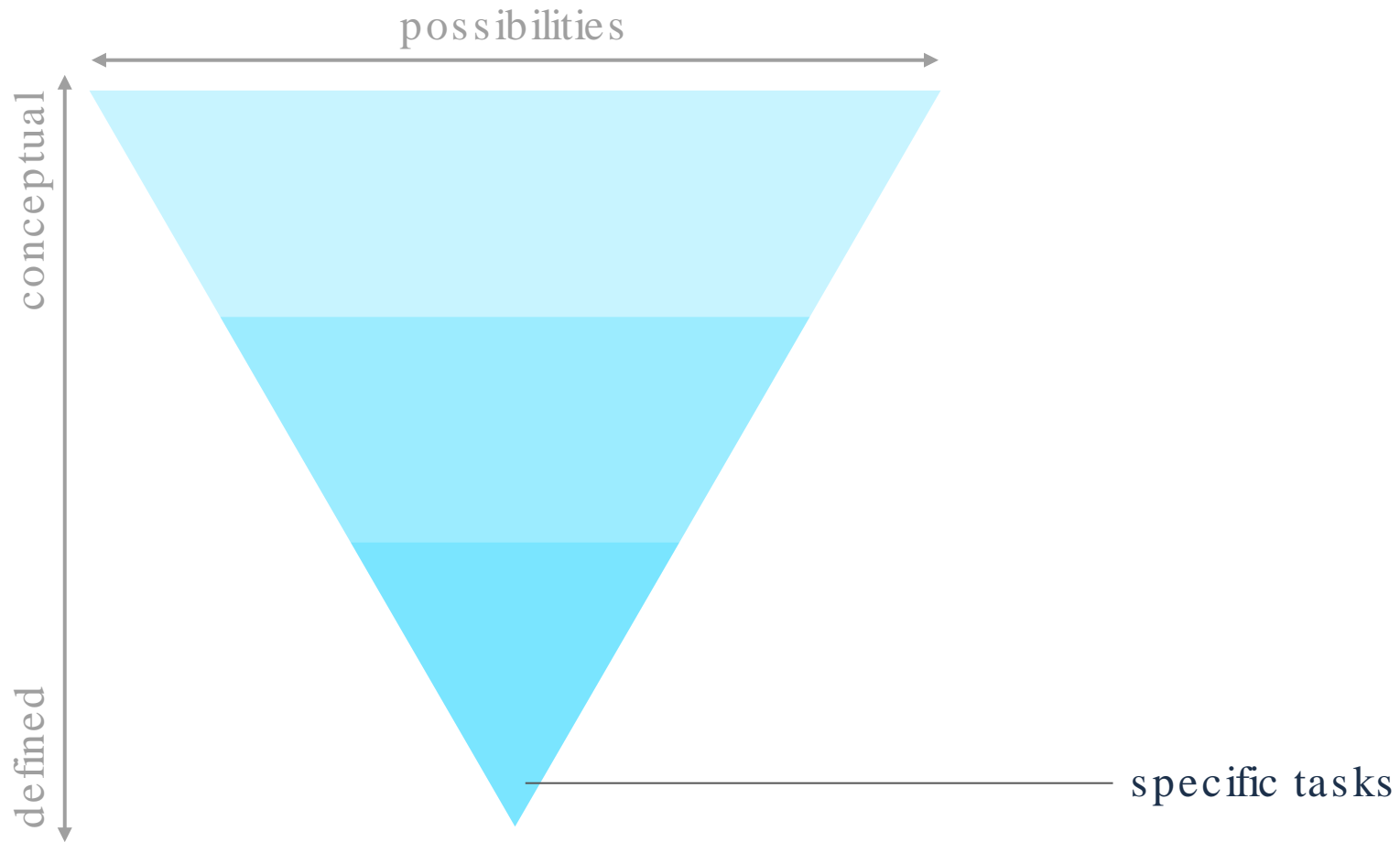


**Roadmaps are not a promise.**

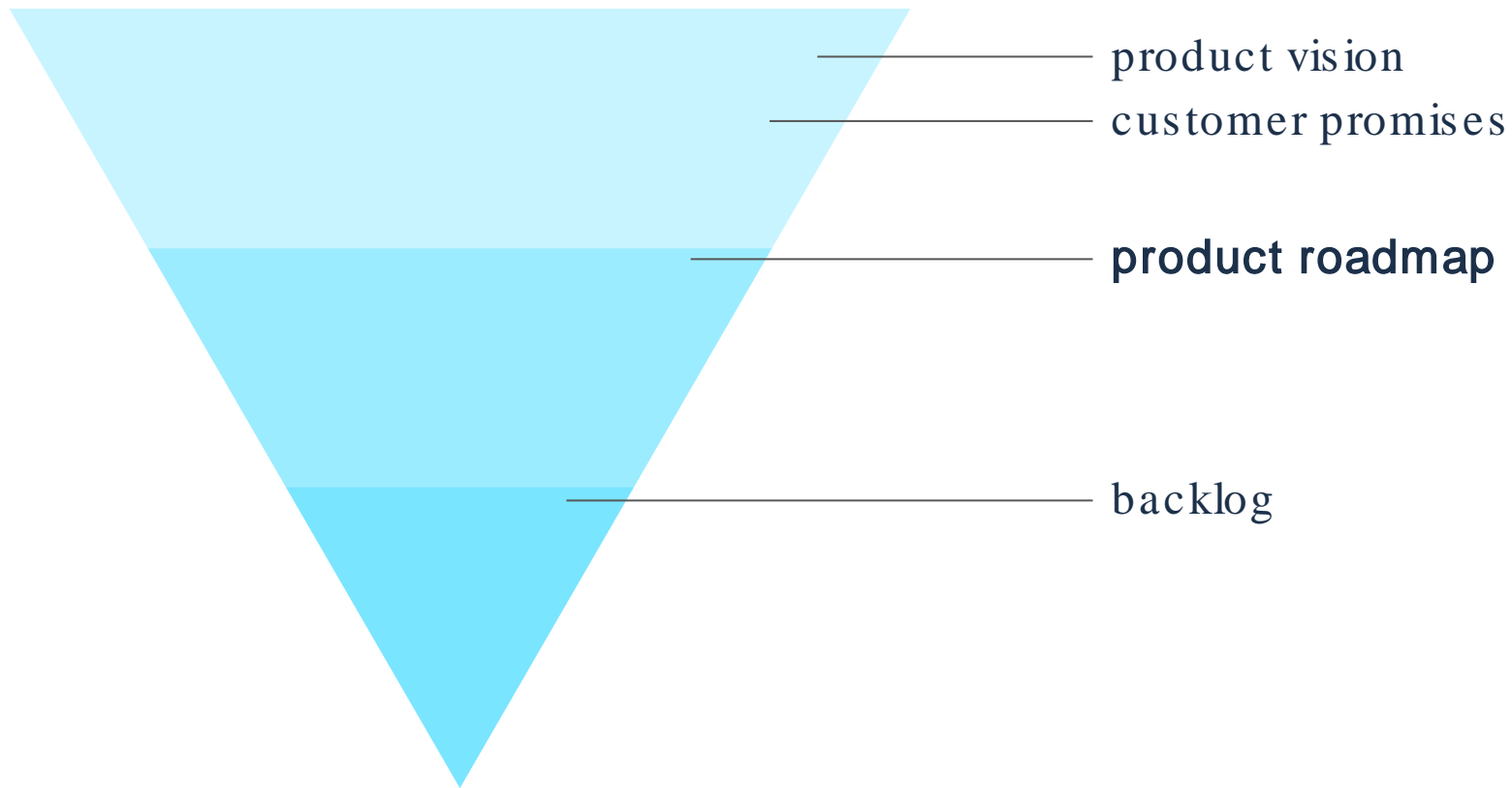
# The spectrum of work

**All of the work of building a product lives on a spectrum — you need open-ended conceptual work just as much as clearly-defined execution tasks.**





**The roadmap is a bridge between  
your strategic vision and your  
backlog of work.**



# Pulling from your customer promises



Customer promises are the backbone of your roadmap. They dictate the **scope** of the team's work.

**The first step to building a roadmap is to put your customer promises into a meaningful order.**

The first step to building a roadmap is to put your customer promises into a meaningful order. **When in doubt, prioritize.**

**Imagine you're building a web portal where users can look up their tickets and traffic violations.**

## Vision

**We're creating a way for members of the public to view and address their tickets and traffic violations, so that traffic incidents are resolved more efficiently.**

# Customer promises

A member of the public can see all of their current unpaid tickets.

A member of the public can see their past tickets and traffic violations.

A member of the public can pay off an unpaid ticket.

A member of the public can initiate the process of contesting a ticket.

Agency staff can track currently unpaid tickets.

Agency staff can see a record of tickets and traffic violations.

# Sequencing customer promises

1

What is the **core value proposition** of this system? Which of these customer promises most directly support it?

2

Which customer promises must be huge successes?  
Which ones just need to get completed?

3

Are there some customer promises that must get fulfilled sooner than others? Which ones, and why?

# Prioritizing customer promises

P1

A member of the public can see all of their current unpaid tickets.

A member of the public can pay off an unpaid ticket.

Agency staff can track currently unpaid tickets.

P2

A member of the public can see their past tickets and traffic violations.

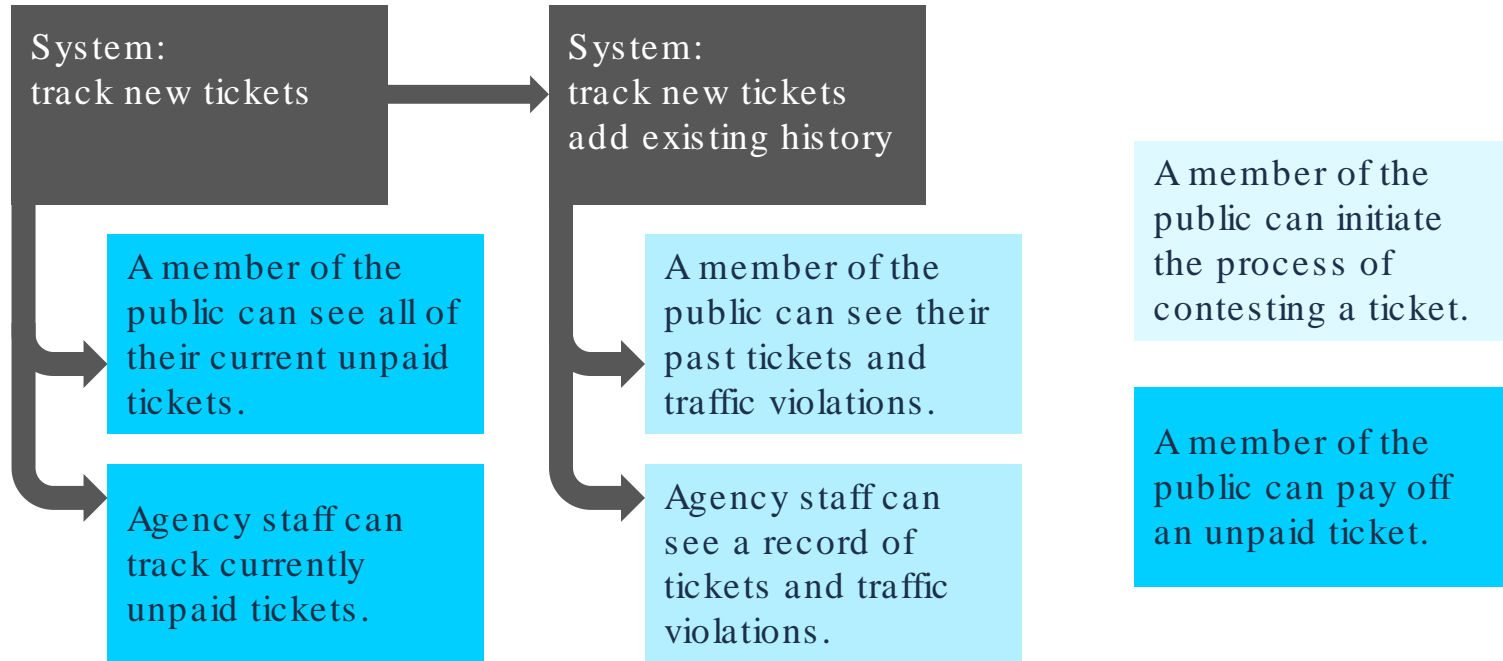
Agency staff can see a record of tickets and traffic violations.

P3

A member of the public can initiate the process of contesting a ticket.



# Sequencing customer promises



**Once you've figured out the priority, and relative timing, of your customer promises, you can put them into sequential order.**

# Discussion

If your team could only work on one customer promise at a time, how would you sequence the work?

\*

There's no single right answer. The important thing is to have a reason why you've chosen a particular sequencing.

# Discussion

If your team work on **two** customer promises at a time, how would you sequence the work?

\*

Remember that you don't know exactly how long each customer promise will take — how will you handle that?

**Let's imagine that our team is pretty small, and can only focus on one customer promise at a time.**

discussion

## Strict priority order

A member of the public can see all of their current unpaid tickets.

Agency staff can track currently unpaid tickets.

A member of the public can pay off an unpaid ticket.

A member of the public can see their past tickets and traffic violations.

Agency staff can see a record of tickets and traffic violations.

A member of the public can initiate the process of contesting a ticket.

discussion

## Prioritizing the public

A member of the public can see all of their current unpaid tickets.

A member of the public can see their past tickets and traffic violations.

Agency staff can track currently unpaid tickets.

Agency staff can see a record of tickets and traffic violations.

A member of the public can pay off an unpaid ticket.

A member of the public can initiate the process of contesting a ticket.

discussion

## Prioritizing the agency

Agency staff can track currently unpaid tickets.

Agency staff can see a record of tickets and traffic violations.

A member of the public can pay off an unpaid ticket.

A member of the public can see all of their current unpaid tickets.

A member of the public can see their past tickets and traffic violations.

A member of the public can initiate the process of contesting a ticket.



**This is already a (rough) roadmap!**

**Note that it doesn't tell you when each customer promise will be completed, or even what the solution will look like.**

Note that it doesn't tell you when each customer promise will be completed, or even what the solution will look like. **It just tells you what order the work will be completed in.**

# Zooming in

A member of the public can see all of their current unpaid tickets.

A member of the public can see their past tickets and traffic violations.

Agency staff can track currently unpaid tickets.

Agency staff can see a record of tickets and traffic violations.

A member of the public can pay off an unpaid ticket.

A member of the public can initiate the process of contesting a ticket.

**Of course, this roadmap is probably still too high level for the team.**

Of course, this roadmap is probably still too high level for the team. **It needs to be broken down into more detailed pieces of work.**

A member of the public can see all of their current unpaid tickets.

A member of the public can see their past tickets and traffic violations.

Agency staff can track currently unpaid tickets.

Agency staff can see a record of tickets and traffic violations.

A member of the public can pay off an unpaid ticket.

A member of the public can initiate the process of contesting a ticket.

track new tickets

user account  
creation and  
management

associate tickets  
with user accounts

display tickets



**Each of these items can get broken into more detailed tasks, too.**

track new tickets

user account  
creation and  
management

associate tickets  
with user accounts

display tickets

create new account

log in/out (session  
management)

associate account  
with legal identity

account recovery  
(password reset)

**And so on, until you have bite-sized pieces of work.**

**Each piece of work connects  
logically to an overarching customer  
promise.**

A member of the public can see all of their current unpaid tickets.



track new tickets

user account  
creation and  
management

associate tickets  
with user accounts

display tickets



create new account

log in/out (session  
management)

associate account  
with legal identity

account recovery  
(password reset)

# Stitching it all together

When you zoom back out, you get an all-up view of your upcoming work, anchored to the **end value** you're delivering.

A member of the public can see all of their current unpaid tickets.

A member of the public can see their past tickets and traffic violations.

Agency staff can track currently unpaid tickets.

Agency staff can see a record of tickets and traffic violations.

A member of the public can pay off an unpaid ticket.

A member of the public can initiate the process of contesting a ticket.

track new tickets



access historical ticket data



user account creation and management



associate tickets with user accounts



associate tickets with user accounts



display tickets



display tickets





A member of the public can see all of their current unpaid tickets.

A member of the public can see their past tickets and traffic violations.

Agency staff can track currently unpaid tickets.

Agency staff can see a record of tickets and traffic violations.

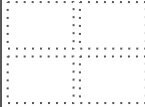
A member of the public can pay off an unpaid ticket.

A member of the public can initiate the process of contesting a ticket.

track new tickets



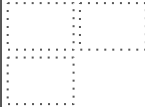
access historical ticket data



user account creation and management



associate tickets with user accounts



associate tickets with user accounts



display tickets



display tickets



Well-understood work is represented in more detail

A member of the public can see all of their current unpaid tickets.

A member of the public can see their past tickets and traffic violations.

Agency staff can track currently unpaid tickets.

Agency staff can see a record of tickets and traffic violations.

A member of the public can pay off an unpaid ticket.

A member of the public can initiate the process of contesting a ticket.

track new tickets

access historical ticket data

user account creation and management

associate tickets with user accounts

associate tickets with user accounts

display tickets

display tickets

Work that comes later is represented in less detail, because it's not well-known yet

A member of the public can see all of their current unpaid tickets.

A member of the public can see their past tickets and traffic violations.

Agency staff can track currently unpaid tickets.

Agency staff can see a record of tickets and traffic violations.

A member of the public can pay off an unpaid ticket.

A member of the public can initiate the process of contesting a ticket.

track new tickets

access historical ticket data

user account creation and management

associate tickets with user accounts

associate tickets with user accounts

display tickets

display tickets

Work that's really far off is just a guess!

A member of the public can see all of their current unpaid tickets.

A member of the public can see their past tickets and traffic violations.

Agency staff can track currently unpaid tickets.

Agency staff can see a record of tickets and traffic violations.

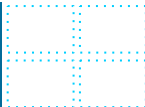
A member of the public can pay off an unpaid ticket.

A member of the public can initiate the process of contesting a ticket.

track new tickets



access historical ticket data



user account creation and management



associate tickets with user accounts



associate tickets with user accounts



display tickets



display tickets



This becomes the backlog your team works off of!

# Scrum overview

# Agenda

1. What is Scrum?
2. Scrum Values
3. Scrum Roles
4. Scrum Ceremonies

# What is Scrum?

**Once you have a product vision, and a roadmap, you need some kind of a framework to help you execute**



**It should do the heavy lifting around communication and planning so that you don't have to manage that manually**

**It should also result in continuous,  
incremental delivery of measurable  
value**

**Scrum** is *one* agile product development framework that is designed to do these things

**The roots of Scrum lie deep in the concept of Lean as it was first laid out for the Toyota Production System (TPS) and introduced as a software methodology in the 1990s**

Product development using Scrum

## Scrum delivery cycle

1

Prioritize

2

Plan

3

Build

4

Ship

5

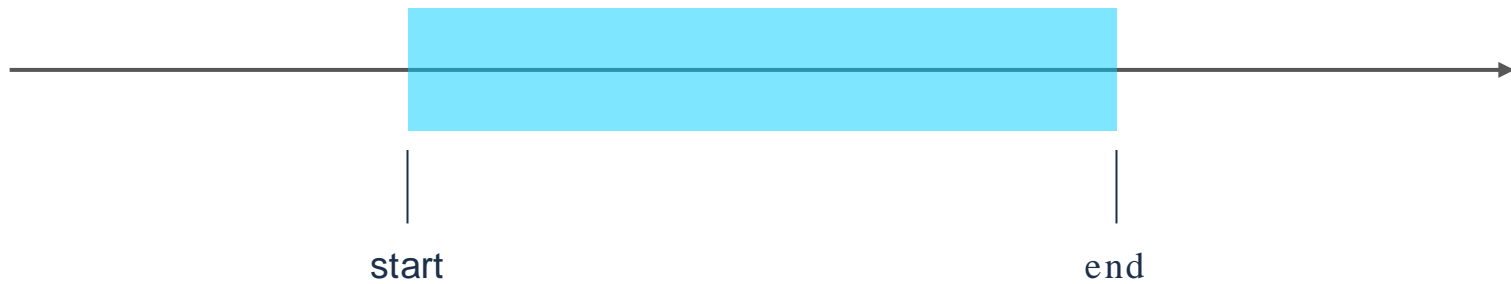
Reflect

**Scrum is focused on delivering quality software rapidly and in small increments.**

**It is also focused on breaking down barriers to communication and optimizing for efficiency.**

**Teams work in time-delimited iterations called sprints.**





## Prioritize

**At the beginning of the sprint, work is scoped at a high level and prioritized into a backlog**

to-do

P1

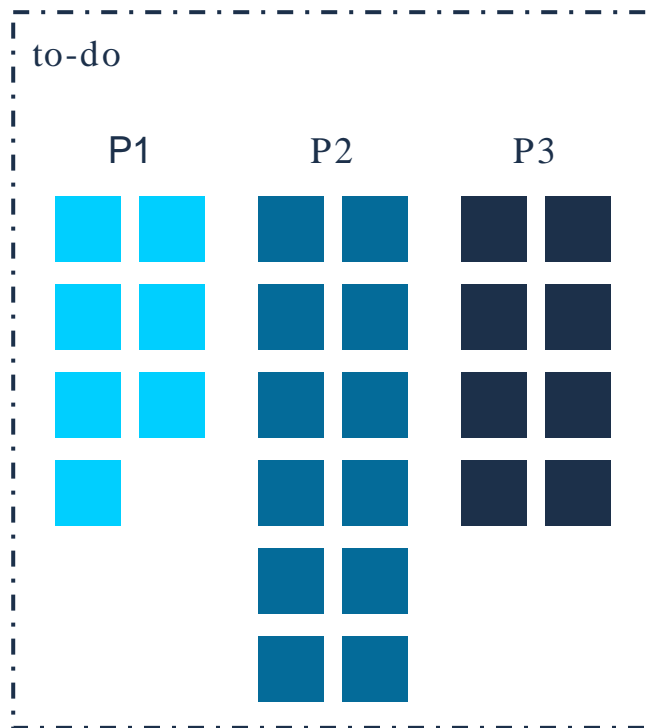
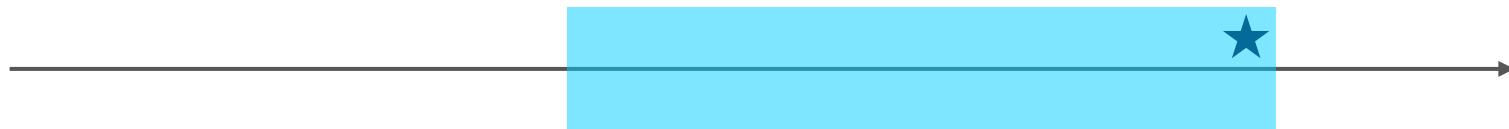
P2

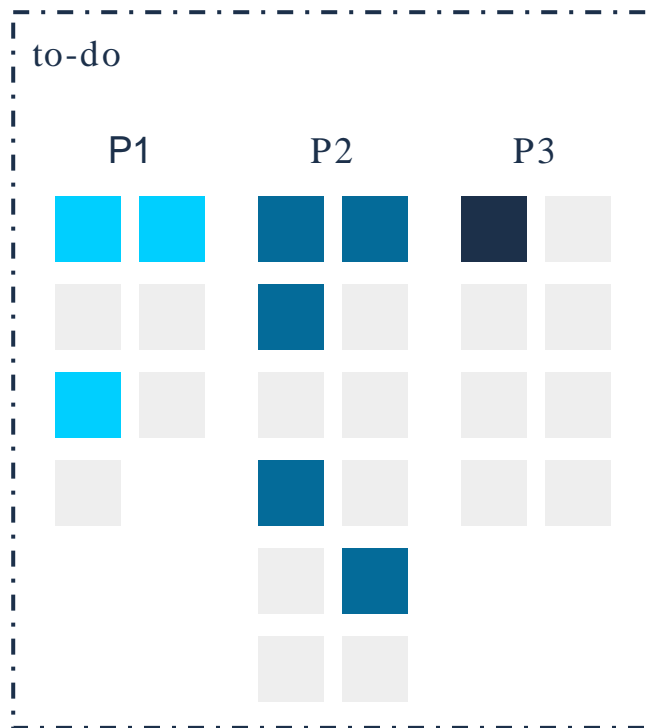
P3



## Plan

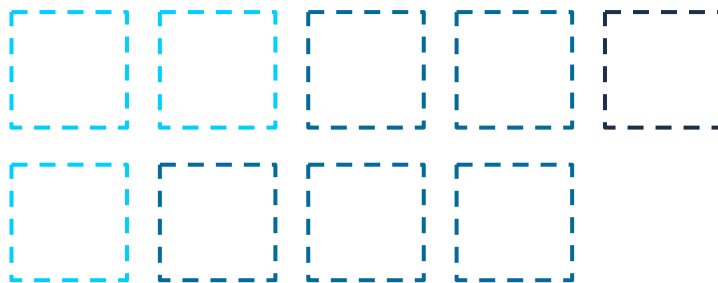
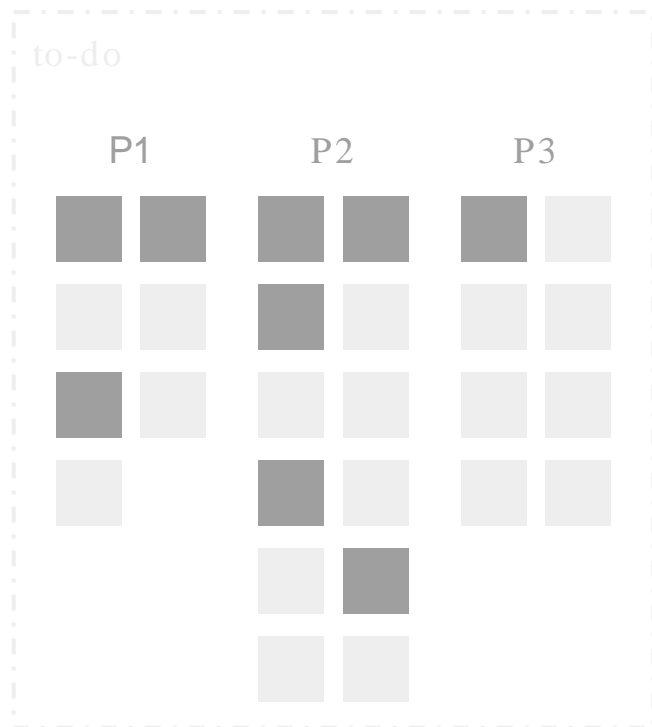
The prioritized work is then further scoped into small tasks that are grouped together for a sprint



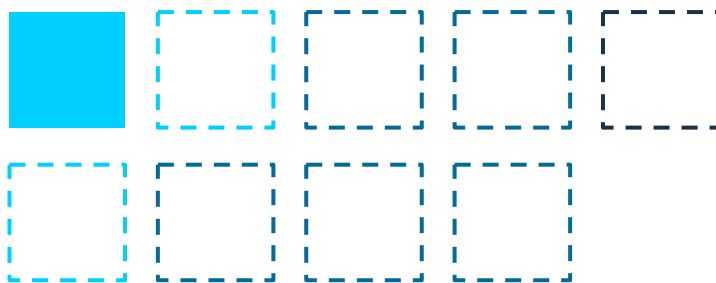
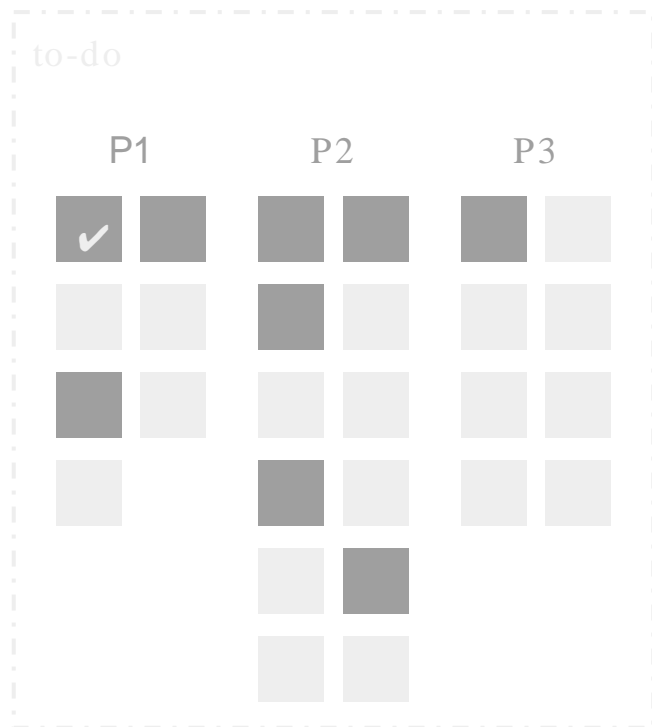


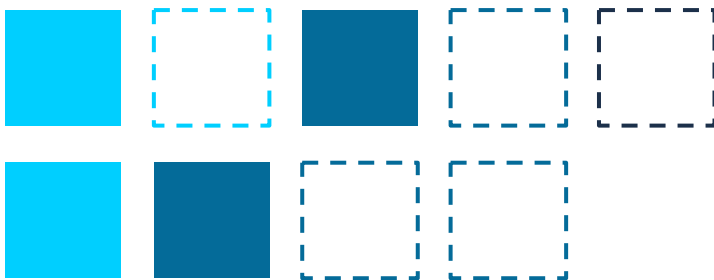
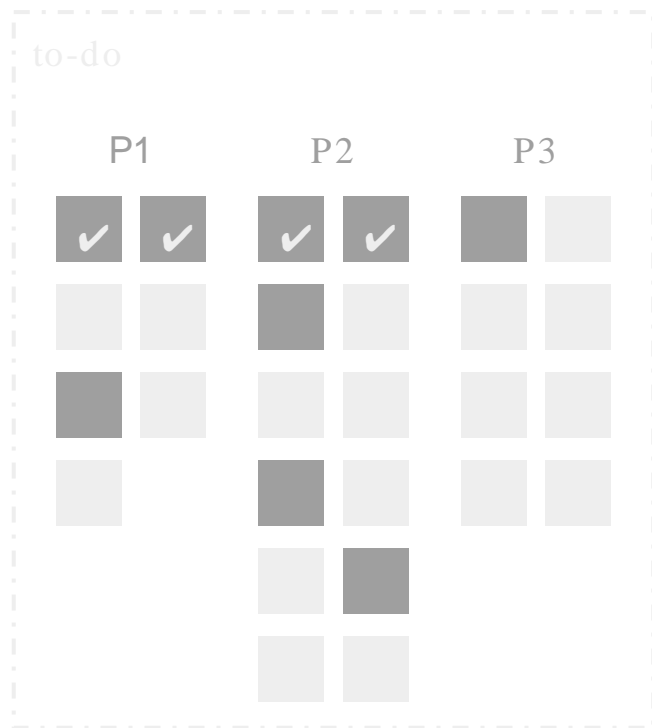
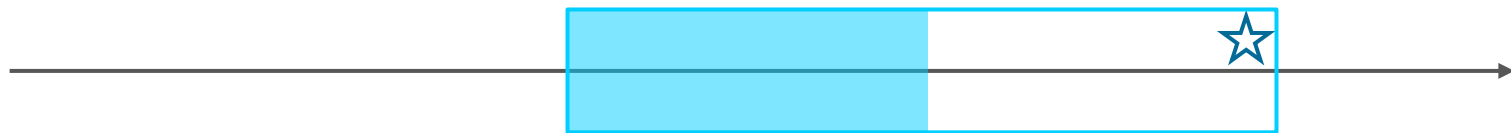
## Build

**During the sprint teams work only on those tasks, and check in daily to retain their focus**



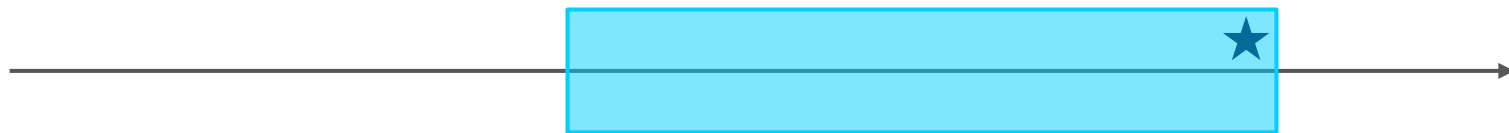






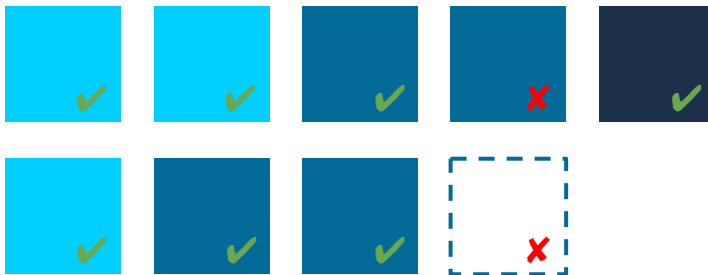
## Ship

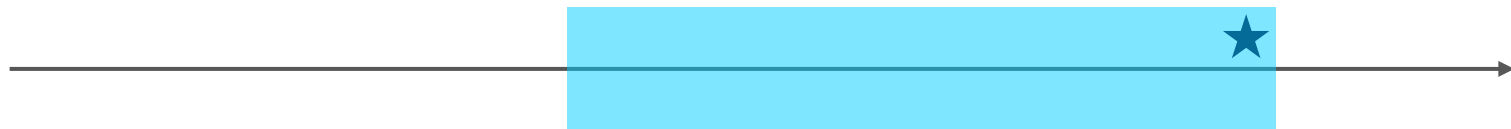
At the end of the sprint, all delivered work is reviewed and the team decides whether it's ready to shipped (deployed to production)



to-do

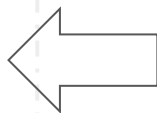
| P1 |   | P2 |   | P3 |  |
|----|---|----|---|----|--|
| ✓  | ✓ | ✓  | ✓ | ✓  |  |
|    |   | ✓  |   |    |  |
| ✓  |   |    |   |    |  |
|    |   | x  |   |    |  |
|    |   |    | x |    |  |
|    |   |    |   |    |  |

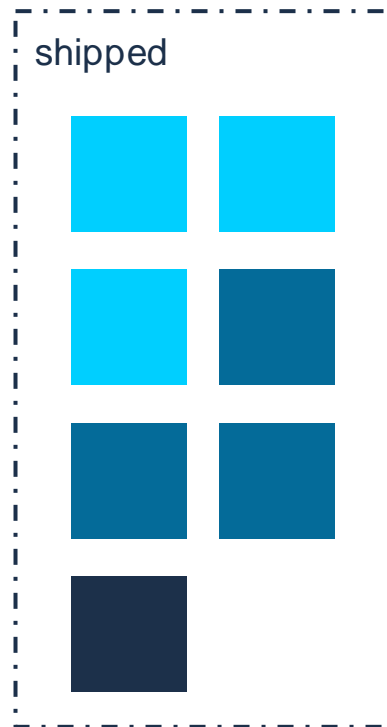
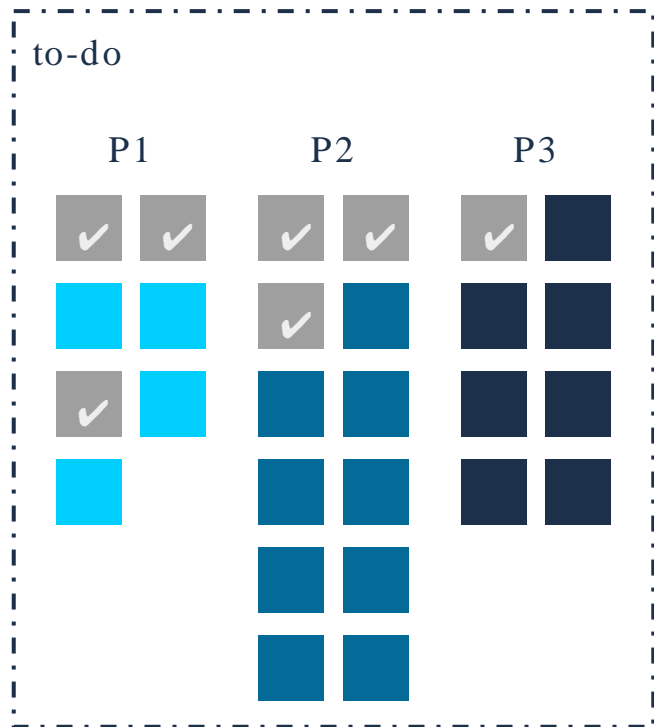
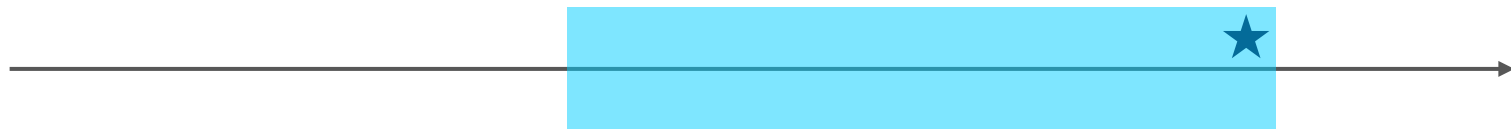




to-do

| P1 |   | P2 |   | P3 |  |
|----|---|----|---|----|--|
| ✓  | ✓ | ✓  | ✓ | ✓  |  |
|    |   | ✓  |   |    |  |
| ✓  |   |    |   |    |  |
|    |   | ✗  |   |    |  |
|    |   |    | ✗ |    |  |
|    |   |    |   |    |  |





## Reflect

At the end of the sprint, we take time to reflect on how we can better deliver value to our users, either through process improvements, or responding to feedback.

**There are lots of variations in the way Scrum is implemented, but they are all based on a core set of values**



# Scrum values

# Focus

**Scrum enables the team to focus on priorities, so that value is delivered quickly**

**We scope work to be small and understandable, so that everyone can align on what to do and why we are doing it**

We aim to plan *just enough* work to get to our sprint goal within the time frame of the sprint

And we focus on *just that work* during the sprint so that we can meet our goal

# Commitment

Commitment is about *dedication* rather than delivering a specific set of features by a deadline.



**We commit to the sprint goal.**

**We commit to focusing on planned work for a given sprint.**

**We commit to communicating and learning throughout the sprint cycle.**

**We commit to doing everything we can to continuously deliver value to our users.**

**We commit to the mission, the level of effort and process, not to specific tasks.**

# Openness

**Teams need to be open to changing strategies or even goals, as users start to give feedback.**

**If the feedback validates our value assumptions, we can continue to build towards it.**



**Sprint 1**



**Sprint 1**

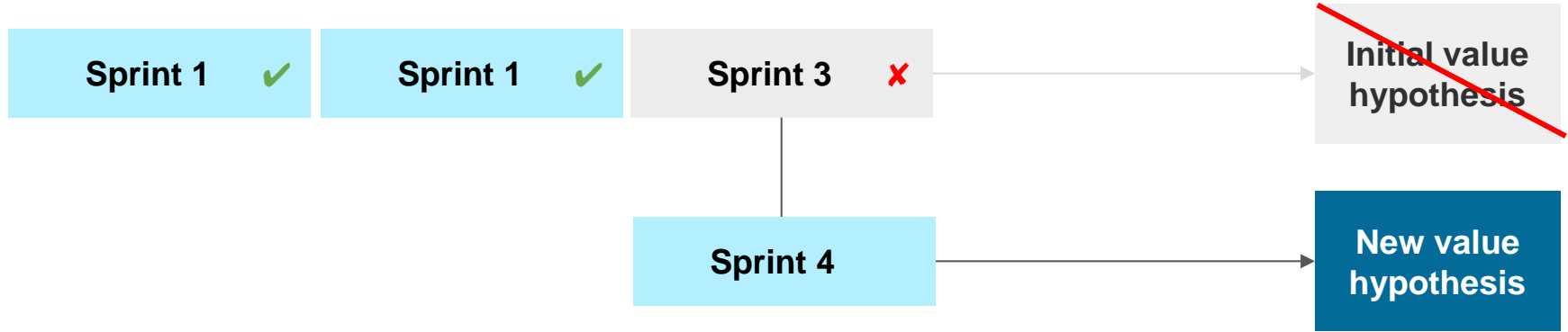


**Sprint 3**



**Initial value  
hypothesis**

**If users tell us we're not delivering value, we change our strategy or goal accordingly.**



# Respect

**We respect our users and respond to them when something is not working.**

**We respect our stakeholders by  
listening and bringing them along.**

**We respect the role of the product owner to hold the vision and decide on priorities.**

**We respect the role of the scrum master to drive the team.**



**We respect the role of designers to advocate for the end user and contribute to the product direction.**

**We respect the role of developers to determine the technical implementation and contribute to product direction.**

# Courage

**We have the courage to realize that planning does not eliminate change and uncertainty.**

**We need to have the courage to admit that requirements will never be perfect and we will fail on occasion.**

**We need the courage to speak about  
our failures so that everyone can learn.**

**We need to have the courage to change course, sometimes dramatically, when faced with data that contradicts our assumptions.**

# Scrum roles



The **Product Owner** holds the vision and directs the priorities of the Product Team

**This single point of direction is critical  
to ensuring the team stays focused**

The **Product Team** is comprised of cross functional subject matter experts that help the Product Owner prioritize, and plan.

They self-organize, with the help of a **Scrum Master**, to get the work done for each sprint.

The **Scrum Master** is a facilitator and coach, ensuring that best practices are being followed by the entire team, and removing any blocks to delivering value

**They need to have a deep understanding of the product goals, so they can work to optimize value delivery through process management**

**They are not project managers. They enable the team to self-organize, rather than driving them to stick to a predetermined schedule.**

# Scrum ceremonies



# Scrum ceremonies

| Ceremony                    | Phase      |
|-----------------------------|------------|
| Backlog refinement          | Prioritize |
| Sprint planning             | Plan       |
| Daily “stand-up” or “scrum” | Build      |
| Sprint review               | Ship       |
| Sprint retrospective        | Reflect    |

**These ceremonies are designed to move the scrum development cycle along.**

**They are intended to facilitate communication and planning, so that the team does not have to do this ad-hoc.**

**They are time-boxed (like sprints) and are dedicated to specific tasks.**

# Best practices

**Use these meetings for the purpose they were intended so the team can focus on getting through each phase in the development cycle.**

**Respect the time-box: if you didn't get to something during the meeting it likely was not a priority.**

**Prepare for Daily Stand-ups by updating the status of your work in your tracking tool (Trello, GitHub, etc), and identifying your blockers.**



**Prepare for Grooming and Planning meetings by reviewing the work in the product backlog prior to the meeting.**

**Prepare for Sprint Reviews by making sure you can demo your work to the rest of the team.**

**Prepare for Retrospectives by thinking about how to improve the team's process, and be open to feedback.**

**Take turns leading meetings so that no one person is a bottleneck.**

# Questions?

**Was this presentation helpful? What else would you like to see in this overview?**

**End of presentation**

# Testing and triage



**As a product manager, you're responsible for setting, and upholding, the standard of quality for your product.**

**Everyone on the team helps find and fix problems.**

Everyone on the team helps find and fix problems. **Product managers are the ones making judgment calls about what to fix and when.**

**Product managers also make sure that there are consistent standards and practices in place.**

**When you're lucky, that means asking the testing lead what process they'd like to use.**

When you're lucky, that means asking the testing lead what process they'd like to use. **The rest of the time, it means helping the team establish a set of habits.**

# The basics of testing

**You can't ensure that your product is doing what it's supposed to if you're not testing it.**



**There are lots of different ways to approach testing – as with many things, there's no One True Approach.**

There are lots of different ways to approach testing – as with many things, there's no One True Approach. **But there are some best practices you can follow.**

**1. QA people are f\*cking awesome**

1. QA people are f\*cking awesome

**2. Your users aren't your QA team**

1. QA people are f\*cking awesome
2. Your users aren't your QA team
- 3. Automation is critical**

1. QA people are f\*cking awesome
2. Your users aren't your QA team
3. Automation is critical
- 4. Automation doesn't catch everything**

1. QA people are f\*cking awesome
2. Your users aren't your QA team
3. Automation is critical
4. Automation doesn't catch everything
- 5. Testing is a qualitative process**

1. QA people are f\*cking awesome
2. Your users aren't your QA team
3. Automation is critical
4. Automation doesn't catch everything
5. Testing is a qualitative process
- 6. Make it a habit, not a production**



**A mix of automated and manual tests should be incorporated into your workflow.**

# Automated testing

**DevOps practices use a lot of automation, with two practices: Continuous Integration and Continuous Deployment.**

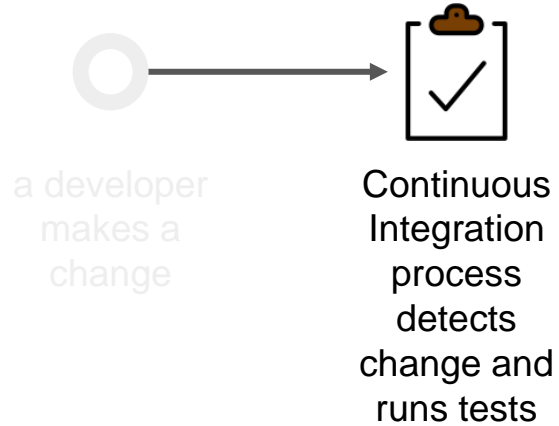
DevOps practices use a lot of automation, with two practices: Continuous Integration and Continuous Deployment. **Automated testing is integral to both of these functions.**

# An automated DevOps process



a developer  
makes a  
change

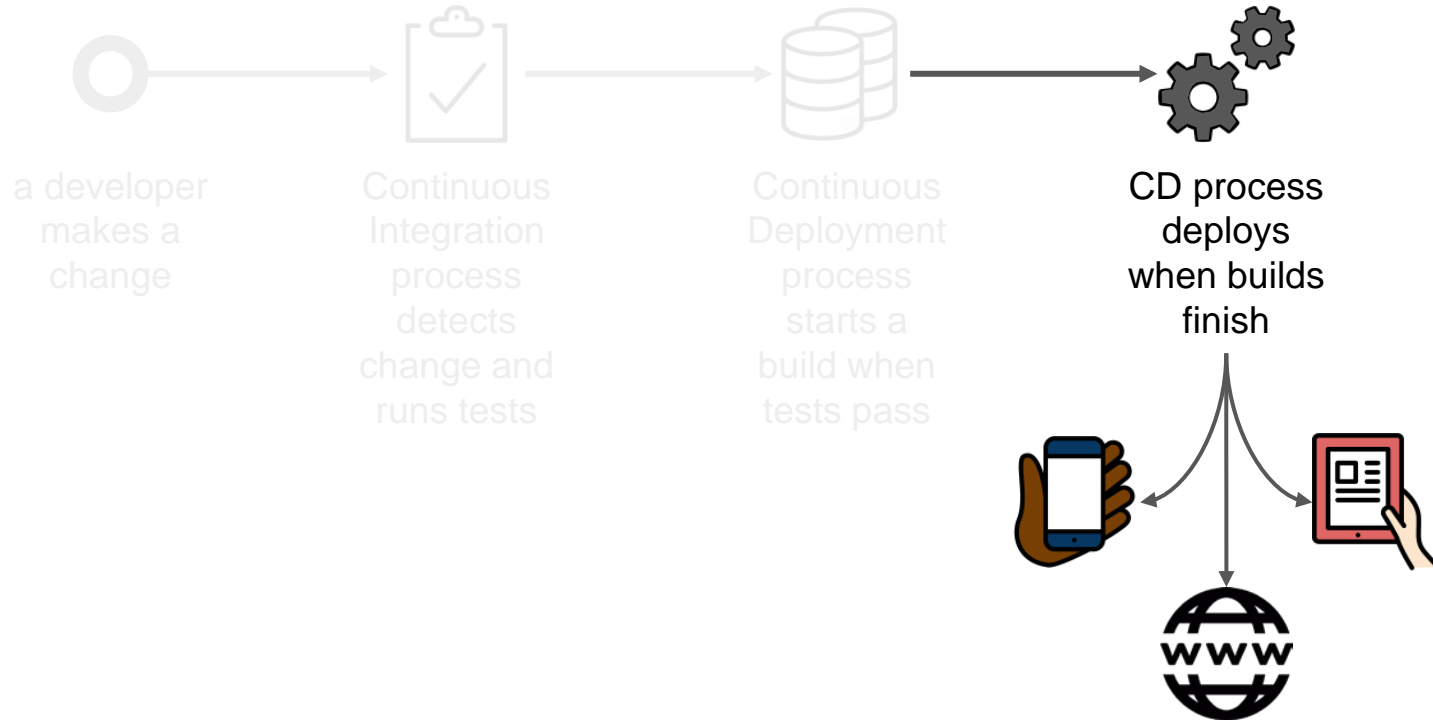
# An automated DevOps process



# An automated DevOps process

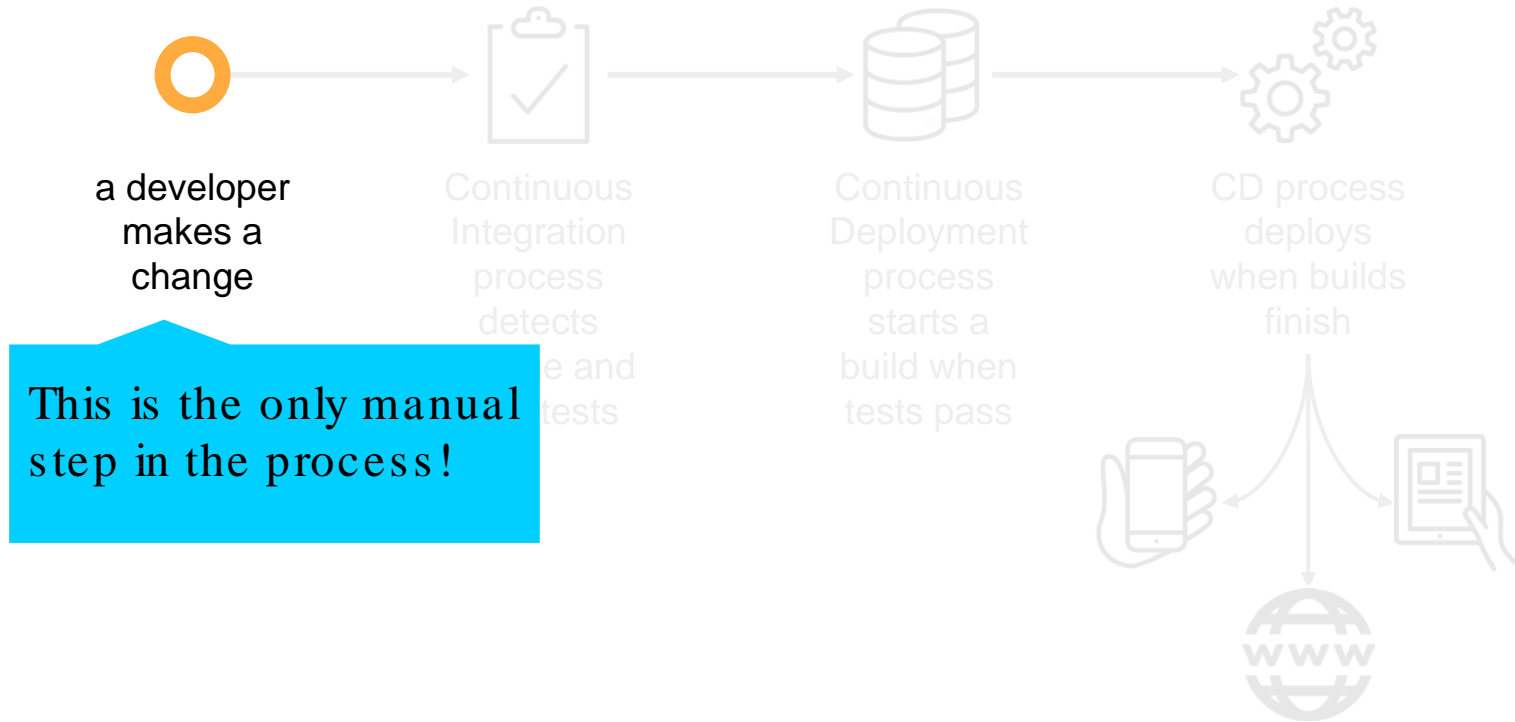


# An automated DevOps process





# An automated DevOps process



# The process of building a backlog

**Turning a vision statement and a roadmap into an actionable backlog is an iterative process.**

**You start with something large and continuously break it into smaller and smaller pieces, until you have something small enough to take action on.**

**The key is to continually focus on user needs and outcomes (rather than specific implementation details)**

milestone

milestone

component

component

component

milestone

component

task  
group

component

component

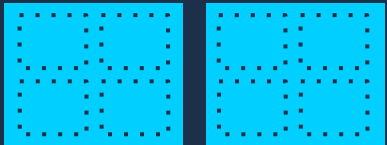
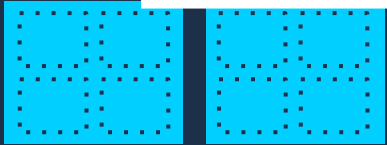


milestone

component



individual tasks



component

component

**The way you build your backlog sets  
the tone for your entire milestone.**

The way you build your backlog sets the tone for your entire milestone. You need to make sure your teammates are actively involved and bought in at every point.

**Once your team has aligned on big picture priorities, pick the first one and consider:**

**Whose needs are you addressing?**

**What are the unknowns?**

**What activities/teams are involved?**

**What outcome(s) are you trying to achieve?**



**How will you know you've succeeded?**

**The act of breaking down the work will help make priorities clearer.**

The act of breaking down the work will help make priorities clearer. **It will also highlight any dependencies between team members, which need to be tracked.**

**Your teammates should be helping you understand, prioritize, and sequence the team's work.**

**Systematically  
breaking it down**

Look at the  
desired  
outcome

Look at the  
desired  
outcome

Talk to the  
team about  
what it takes  
to get there

Look at the  
desired  
outcome

Talk to the  
team about  
what it takes  
to get there

Identify  
components  
of your  
outcome



Look at the  
desired  
outcome

Talk to the  
team about  
what it takes  
to get there

Identify  
components  
of your  
outcome

**Prioritize and  
sequence the  
components**

Look at the  
desired  
outcome

Talk to the  
team about  
what it takes  
to get there

Identify  
components  
of your  
outcome

Prioritize and  
sequence the  
components

Decide what's  
in and what's  
out for this  
cycle

Look at the  
desired  
outcome

Talk to the  
team about  
what it takes  
to get there

Identify  
components  
of your  
outcome

Prioritize and  
sequence the  
components

Decide what's  
in and what's  
out for this  
cycle

For everything  
that's in...

Look at the  
desired  
outcome

Talk to the  
team about  
what it takes  
to get there

Identify  
components  
of your  
outcome

Prioritize and  
sequence the  
components

Decide what's  
in and what's  
out for this  
cycle

For everything  
that's in...



Look at the  
desired  
outcome

Talk to the  
team about  
what it takes  
to get there

Identify  
components  
of your  
outcome

Prioritize and  
sequence the  
components

Decide what's  
in and what's  
out for this  
cycle

**And so on...  
until you have bite-sized tasks.**

For everything  
that's in...

Imagine your team as a drain pipe: the larger the object you try to push through it the more likely it is that blockages and bottlenecks will occur. **Bite-sized tasks keep the flow going.**

**Keep the team focused on the goal.  
Don't let them get lost in the details of  
implementation, so that you can  
quickly deliver value to your users.**

**How do I know when something is  
bite-sized?**



**Reasonably sized**

**The work is relatively small and well understood.**

## Independent

**The work is self-contained, such that it can be worked on by itself.**

## Negotiable

Implementation is flexible, without a specific approach defined.

## Valuable

**The team understands who the user is and how this work provides value to them.**

## Verifiable

The team has a clear idea of what it takes to do the work.

**There will be larger and smaller units of bite-sized work.**

There will be larger and smaller units of bite-sized work. **The goal is to scope each task to the smallest chunk of understandable work that still provides clear value to users**

# An example



**Let's say you want to make it easier for the public to identify affordable healthy food options in their area**

Public can find affordable food options in their area

Define the  
outcome

Public can find  
affordable food  
options in their  
area

Ask the team how  
to get there

Desired outcome:  
Public can find  
affordable food  
options in their area

Technology that  
lets users search  
for food options

Identify  
components of  
the outcome

Desired outcome:  
Public can find  
affordable food  
options in their area

What it takes to get  
there:  
Technology that lets  
users search for food  
options

Database, search  
tool, suggestion  
engine

Desired outcome:  
Public can find  
affordable food  
options in their area

What it takes to get  
there:  
Technology that lets  
users search for food  
options

Components:  
Database of healthy  
food sources, search  
tool, suggestion  
engine

Prioritize and  
sequence work

1. Collect data
2. Build database
3. Search tool
4. Suggestion engine

Desired outcome:  
Public can find  
affordable food  
options in their area

What it takes to get  
there:  
Technology that lets  
users search for food  
options

Components:  
Database of healthy  
food sources, search  
tool, suggestion  
engine

Priority and  
sequence:  
1. Collect data, 2.  
Build database, 3.  
Search tool, 4.  
Suggestion engine

Decide what's in  
and out this cycle

1. Collect data  
2. Build database  
3. Search tool  
~~4. Suggestion  
engine~~

People can find affordable food options

Database

Search tool

Suggestion  
engine



Desired outcome:

People can find  
affordable food  
options

What it takes to get  
there: Technology  
that lets users search  
for food options

Components:  
Database of healthy  
food sources, search  
tool, suggestion  
engine

Priority and  
sequence: 1. Collect  
data, 2. Build  
database, 3. Search  
tool, 4. Suggestion  
engine

1. Collect data  
2. Build database  
3. Search tool  
~~4. Suggestion  
engine~~

In order to  
collect data  
for the  
database...

Define the  
outcome

Comprehensive  
data source for  
healthy food to  
provide good  
information

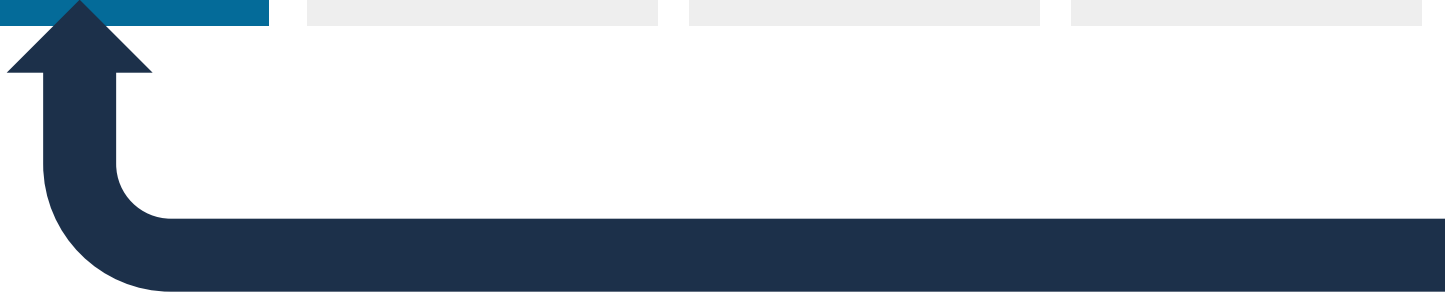
Talk to the  
team about  
what it takes  
to get there

Identify  
components  
of your  
outcome

Prioritize and  
sequence the  
components

Decide what's  
in and what's  
out for this  
cycle

In order to  
collect data  
for the  
database...



Ask the team how  
to get there

Desired outcome:  
Comprehensive data  
source for healthy  
food so the system  
can provide valuable  
information to users

A good sense of  
what data is  
available and a  
quality ranking

Identify  
components of  
the outcome

Desired outcome:  
Comprehensive data  
source for healthy  
food so the system  
can provide valuable  
information to users

What it takes to get  
there:  
A good sense of  
what data is available  
and a quality ranking

Research around  
sources of data, a  
rubric to assess  
quality

Desired outcome:  
Comprehensive data  
source for healthy  
food so the system  
can provide valuable  
information to users

What it takes to get  
there:  
A good sense of  
what data is available  
and a quality ranking

Components:  
Research around  
sources of data, a  
rubric to assess  
quality

Prioritize and  
sequence work

1. Research
2. Document
3. Develop rubric  
to assess quality

Desired outcome:  
Comprehensive data  
source for healthy  
food so the system  
can provide valuable  
information to users

What it takes to get  
there:  
A good sense of  
what data is available  
and a quality ranking

Components:  
Research around  
sources of data, a  
rubric to assess  
quality

Priority and  
sequence:  
1. Research, 2.  
Document 3. Develop  
rubric to assess  
quality

Decide what's in  
and out this cycle

1. Research  
2. Document  
~~3. Develop rubric  
to assess quality~~

People can find affordable food options

Database

Research

Search tool

Suggestion  
engine

Desired outcome:  
Comprehensive data  
source for healthy  
food so the system  
can provide valuable  
information to users

What it takes to get  
there:  
A good sense of  
what data is available  
and a quality ranking

Components:  
Research around  
sources of data, a  
rubric to assess  
quality

Priority and  
sequence:  
1. Research, 2.  
Document 3. Develop  
rubric to assess  
quality

1. Research  
2. Document  
~~3. Develop rubric  
to assess quality~~

In order to do  
research...



Define the  
outcome

Enough research  
to be confident  
that the info will  
help people find  
good options


Talk to the  
team about  
what it takes  
to get there

Identify  
components  
of your  
outcome

Prioritize and  
sequence the  
components

Decide what's  
in and what's  
out for this  
cycle

In order to **do**  
**research...**



```
graph LR; A[Define the outcome] --> B[Talk to the team about what it takes to get there]; B --> C[Identify components of your outcome]; C --> D[Prioritize and sequence the components]; D --> E[Decide what's in and what's out for this cycle]; F[In order to do research...] --> A;
```

**And, so on ...**

**Until you get down to a small unit:  
Research healthy food options in  
Austin, TX**

## Reasonable size

Relative to other work in the queue the team estimates this is a medium sized task

**Independent**

**Limited to Austin, TX with no  
dependencies on other cities**

## Negotiable

The research method isn't specified;  
just the outcome

## Valuable

Understanding the food options in Austin, TX will allow us to help people there find healthy food options

## Verifiable

**A list of healthy food options in Austin,  
including location and type of food**



People can find affordable food options

People can find affordable food options

Database

Search tool

Suggestion  
engine

People can find affordable food options

Database

Research

Search tool

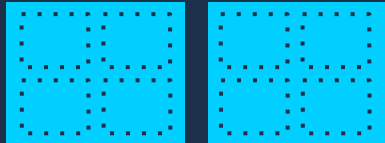
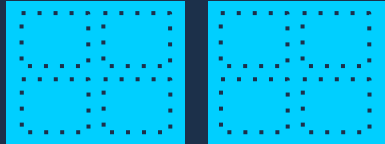
Suggestion  
engine

People can find affordable food options

Database



Austin, TX



Search tool

Suggestion  
engine

**Tips for splitting tasks**

**<http://agileforall.com/wp-content/uploads/2012/01/Story-Splitting-Flowchart.pdf>**

# Practice

**Let's break down one of your  
milestones**